

See you at Transportation Camp!



All roads (and rails, sharrows, and BRT corridors) lead to Washington, D.C. this month—if you’re a transit nerd, that is. There’s the **Transportation Camp** (<http://transportationcamp.org>) unconference tomorrow, the **Transportation Research Board** (<http://www.trb.org/AnnualMeeting2015/AnnualMeeting2015.aspx>)’s annual week-long meeting next week, and dozens of happy hours squeezed in between. Mapzen will be there, and we’d enjoy the chance to meet you, if you happen to be there too.

Transportation Camp attracts participants from a wide variety of fields and interests, from urban planners and transportation engineers to journalists and software engineers. We can’t wait to talk transit data production, collection, and formatting with such a diverse group. If you’re in D.C. and are interested in transit, Mapzen or both, come find **me** (<https://twitter.com/meghanhade>), **Drew** (<https://twitter.com/drewdaraabrams>), **Ekta** (<https://twitter.com/ektadary>), or **Alyssa** (<https://twitter.com/alyssapwright>).

Along with our friends at **Conveyal** (<http://conveyal.com>) and **TransitScreen** (<http://transitscreen.com>), we’ll be co-hosting a post-Transportation Camp and pre-TRB happy hour this Sunday evening. We’d love to see you there! More information and RSVP **here** (<http://conveyaldctrbparty.splashthat.com/>).

· 08 January 2015 ·



Meghan Hade

Meghan is a software engineer with a background in urban planning on Mapzen's Mobility team. She works in javascript and plays in calligraphy, block printing, and finding ways to stash n+1 bikes in a San Francisco apartment.

© 2017 Mapzen

Setting up a Pelias Instance

Update: *This post was written a while ago, and many of these instructions for setting up Pelias are out of date. Please see our new **installation instructions** (<http://pelias.io/install.html>), which are being actively updated.*

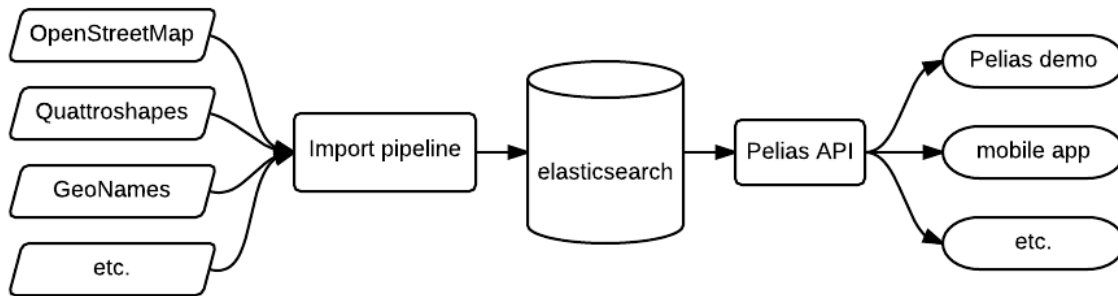
For the past few months, Mapzen has been hard at work developing **Pelias** (<https://github.com/pelias/pelias>), a lightweight, modular implementation of a geocoder that's easy for others to set up on top of their own datasets: everything from proprietary data bundles to OpenStreetMap planet dumps. Pelias is now fairly mature and approaching feature completeness, to the point that you can — **wait for it** — build your very own local instance. Here's how. In this post, we'll briefly take a look at the inner workings of geocoders in general, Pelias specifically, and then build Pelias on top of an **OpenAddresses** (<http://openaddresses.io/>) address dataset.

How it works

A geocoder consists of two fundamental components. The first is an API sitting on top of a database containing reams of geographic data, like records matching longitude and latitude points to textual names, that responds to requests and implements all kinds of search logic on top of what the underlying data-store already provides. The second component is the collection of data pipelines used to normalize, filter, and import geographic data into that database. Pelias relies on **elasticsearch** (<http://www.elasticsearch.org/>), an easily scalable database optimized for text search, and provides **an API** (<https://github.com/pelias/pelias#im-a-developer-can-i-get-access-to-the-api>) with three primary endpoints:

- **reverse geocoding**, or matching a name to a point
- **search** (forward geocoding), or matching a point to a name
- **suggest**, or query suggestion on the fly that tries to inference the text the user might want to search for

We're developing import pipelines for numerous popular open datasets, described in detail below.



Existing importer

Pelias is dataset agnostic, meaning that it will work with any geographic data as long as it's massaged into the proper format. Mapzen is actively developing importers for the following popular datasets:

- **OpenStreetMap** (<https://github.com/pelias/openstreetmap>), a global address/street/region dataset
- **OpenAddresses** (<https://github.com/pelias/openaddresses>), a steadily growing aggregation of normalized municipal address datasets
- **TIGER** (<https://github.com/pelias/tiger>), a street survey of the United States published by the US Census Bureau
- **GeoNames** (<https://github.com/pelias/geonames>), a global point-of-interest dataset
- **Quattroshapes** (<https://github.com/pelias/quattroshapes-pipeline>), a global polygon dump at various levels of granularity.

Custom importer

Since the import pipeline is simply a collection of filters and mappers that accept any input data, it's conceivable that you can write your own importer for any dataset. Our long term goal is to make that process as painless as possible to allow developers to adapt Pelias to their custom datasets. It's currently a fairly complex piece of machinery overall that might, at any given point, perform anything from filtration to address de-duplication, and is being constantly refined and improved (we're working on it, and we welcome pull requests!). For this post, we'll use the dataset-specific importers listed above.

Setting up your own

Pelias has lots of moving parts, but most of the installation and dataset-import process has been abstracted away into a convenient **Vagrant image** (<https://github.com/pelias/vagrant>) that we'll use in this post. It'll build a virtual machine, install and run elasticsearch, the Pelias API, all other minor dependencies, tweak various environment settings, and can optionally import certain datasets for you right out of the box.

Installation

Pelias ships with a Vagrant image that will create a virtual machine (essentially an operating system running inside another operating system) independent of your host computer, and install Pelias in a complete sandbox. That means we can rely on identical working environments, exact library/package versions, and the presence of all necessary dependencies. The image contains **exhaustive documentation**

(<https://github.com/pelias/vagrant/blob/master/README.md>) of all its configurable bells and whistles, but here are the basics. Note that we recommend a box with ~4GB RAM and ~20GB of free disk space, and realize that **the documentation will be up to date. This article will not**. If something is consistently going wrong, check out the docs to ensure that no new dependencies, requirements, or steps were introduced.

Vagrant dependencies

The only dependencies that you'll have to install are those of **Vagrant** (<https://www.vagrantup.com/>) itself. If you're on Ubuntu or a similar Linux distribution, your package manager should suffice for two of the three: `apt-get install virtualbox vagrant` ; if not, install them manually from **VirtualBox** (<https://www.virtualbox.org/wiki/Downloads>) and **Vagrant** (<https://www.vagrantup.com/downloads.html>). Lastly, install **ChefDK** (<https://downloads.chef.io/chef-dk/>).

Building the image

Update: *This post was written a while ago, and many of these instructions for setting up Pelias are out of date. Please see our new **installation instructions** (<http://pelias.io/install.html>), which are being actively updated.*

Once you've successfully installed the above dependencies:

```
git clone https://github.com/pelias/vagrant
cd vagrant
vagrant plugin install vagrant-berkshelf
vagrant plugin install vagrant-omnibus
```

You're now ready to begin building the image, but first we'll need to configure the datasets so that it automatically imports once everything's set up. Create a new file called `pelias_settings.rb` in the root of the Vagrant repo (should be your current directory), and add the following:

```
Vagrant.configure('2') do |config|
  config.vm.provision :chef_solo do |chef|
    chef.cookbooks_path = 'cookbooks'

    chef.json = {
      'pelias' => {
        'schema' => {
          'create_index' => true
        },
        'osm' => {
          'index_data' => true,
          'extracts' => {
            'new-york' => 'https://s3.amazonaws.com/metro-extracts.mapzen.com/new-york_ne
          }
        }
      }
    }

    chef.run_list = [
      'recipe[pelias::default]'
    ]
  end
end
```

This configuration file, based closely on **this more complete version** (https://github.com/pelias/vagrant/blob/master/pelias_settings.example.rb), will instruct Vagrant to perform certain actions while building (like `'create_index' => true`, which creates the Pelias elasticsearch index), and *optionally* import some datasets — in this case, the OpenStreetMap data for New York. You can change the link to any other OSM PBF, like the ones we provide for numerous other cities as **metro extracts** (<https://mapzen.com/metro-extracts/>), or forgo that import entirely by setting `'index_data'` to `false`, like so:

```
'osm' => {  
  'index_data' => false,  
  'extracts' => {  
    'new-york' => 'https://s3.amazonaws.com/metro-extracts.mapzen.com/new-york_new-y  
  }  
}
```

Since we're only importing OpenAddresses data, we're going to skip any OSM imports for now.

Finally, the one command to rule them all: `PELIAS_VAGRANT_CFG="$(pwd)/pelias_settings.rb"`
`vagrant up`. Make sure that you have a stable Internet connection before running it because the bootstrapping process will download *a lot* of software.

Go grab a coffee, since this will take anywhere between 10 and 20 minutes. The generous logging will make it obvious if something breaks along the way, in which case you should simply try running it again (I know, but *trust me*). Errors related to timed-out requests are often the results of a slow network or unresponsive servers, in which case you should wait a bit before pulling the trigger. If you're still experiencing problems, **raise an issue** (<https://github.com/pelias/vagrant/issues/new>).

Once the build completes, give the output a cursory glance to ensure that there weren't any (non-fatal) errors along the way. First, as a sanity check, run `curl localhost:3100` and expect to see `{"name":"pelias-api","version":{"number":"0.0.0"}}`. Then run `vagrant ssh`, and you'll find yourself inside the Pelias box.

Importing a dataset

Now, we'll import a small OpenAddresses dataset. Start by downloading the OpenAddresses bundle to the Vagrant box (this means you still need to be `ssh` 'd in), which consists of a large number of CSV files containing addresses for entire countries, states, and cities at a time.

```
wget http://s3.amazonaws.com/openaddresses/openaddresses-processed.zip  
mkdir openaddresses-data  
unzip -d openaddresses-data openaddresses-processed.zip  
mkdir my_dataset  
cp openaddresses-data/us-ny-nyc.csv my_dataset
```

The newly created `my_dataset` directory will contain all of the OpenAddresses files we'll import into Pelias; `us-ny-nyc.csv` (New York City) was chosen to get you started, but feel free to use any other file or copy more of them into the directory. Then, install the OpenAddresses importer:

```
git clone https://github.com/pelias/openaddresses
cd openaddresses
npm install
node import.js ../my_dataset
```

The last command will kick off the import, so sit back and watch the stats scroll by. (You can get another cup of coffee if you feel like it.) You should see something like the following:

```
{
  "start": 1421159940127,
  "paused": false,
  "transient": 0,
  "current_length": 154,
  "end": 1421159956764,
  "indexed": 500,
  "batch_ok": 1,
  "batch_retries": 0,
  "failed_records": 0,
  "openaddresses": 500,
  "persec": 0
}
```

Pay attention to the `indexed` value: this is the number of documents successfully indexed in elasticsearch.

See it on a map

An API on its own is all well and good, but most users will want to use it in conjunction with something like...*a map!* Our **demo** (<https://github.com/pelias/demo>) provides just that, and, for convenience, we provide a version that ties into the Pelias that you have running *locally*: **check it out** (<http://rawgit.com/pelias/demo/localhost/index.html>). It'll query

`localhost:3100`, or the API running inside your Vagrant image, meaning that you can scroll over to whichever geographic area you imported and poke around. Happy geocoding!

· 14 January 2015 ·



Severyn Kozak

© 2017 Mapzen

Finding your way with LOST

mobile (/tag/mobile) **android** (/tag/android)

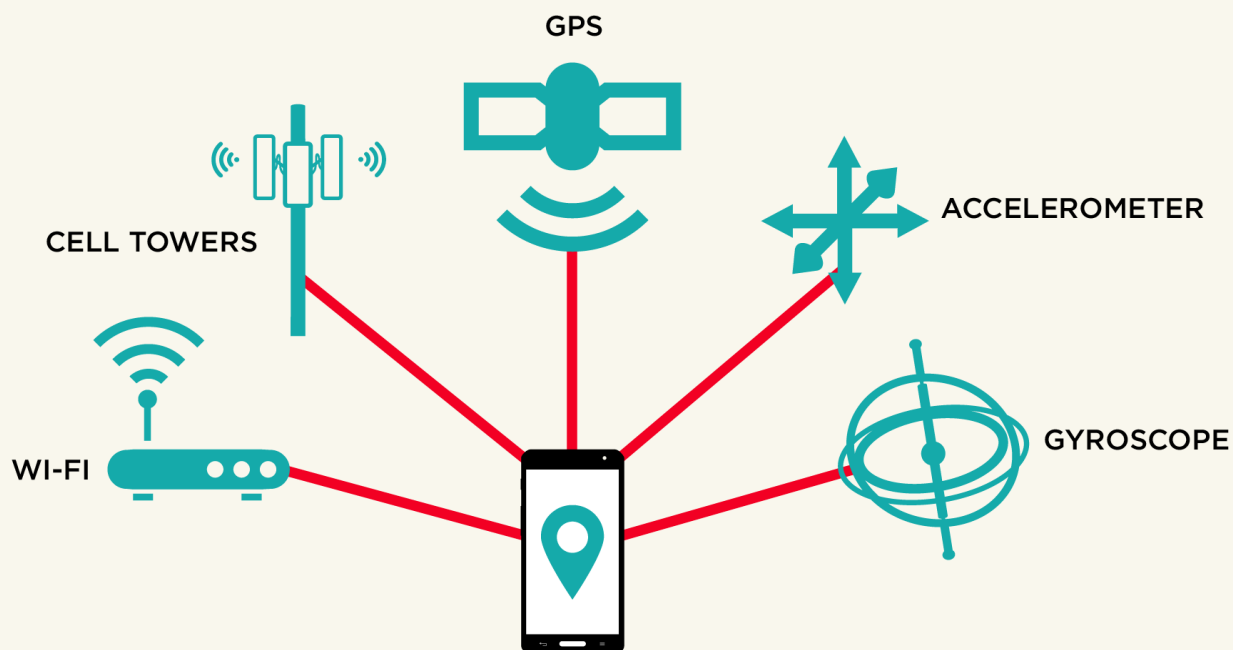
“Not all those who wander are lost.” — J.R.R. Tolkien, The Fellowship of the Ring

Location Open Source Tracker (LOST) (<https://github.com/mapzen/lost>) is an Android SDK we’ve been working on that provides open source location services for Android. It is a drop-in replacement for the **Google Play Services Location APIs** (<https://developer.android.com/google/play-services/location.html>) that depends only on the Android framework.

In non-Android speak: LOST makes it easy for an app on your Android device to figure out where you are, without having to talk to Google about it.

Let’s take a look at how devices figure out where you are, how Google does that, and how LOST does that.

Location Providers



Icon credits: Marie-Louise Merser, Michael Thompson, Kevin Schumacher, Edward Boatman, Naomi Atkinson, Ilur Aptukov from the Noun Project

A location provider is a component, sensor, or system that can be used to determine the geographic location of a device. All location providers have different strengths and weaknesses, so they're usually combined to generate the most accurate location possible. The Android platform uses several location providers:

GPS

The GPS location provider uses satellites to determine location. It has the best accuracy of all the providers but also the highest power consumption. GPS requires an unobstructed line of sight to four or more satellites, which means it is adversely affected by things like tree cover, bridges, tunnels, and buildings.

Network









The network location provider determines location using Wi-Fi and/or cell towers. Wi-Fi positioning operates reasonably well in areas with a high number of visible networks. Cell tower positioning is the least accurate of all location providers but also consumes the least power.

Passive

The passive provider is not a true location provider but rather a low-power alternative to actively requesting a location. Whenever another app or service requests a location update the same location is forwarded to the passive provider with no additional power consumption.

Sensors

Sensors are unable to determine an absolute location by themselves but can be used in conjunction with other providers. Raw data from the accelerometer, gyroscope, and magnetometer can be used to enhance location accuracy and measure displacement from the last known location.

	 GPS	 WiFi	 Cell	 Sensors
Power				
Accuracy				
Coverage				

Source: <https://developers.google.com/events/io/sessions/325337477>
(<https://developers.google.com/events/io/sessions/325337477>)

For more information on location providers check out the **Location and Sensor API Guide** (<https://developer.android.com/guide/topics/sensors/index.html>).

Fused Location Provider

Managing so many different location providers while maximizing accuracy and minimizing power consumption can greatly increase the complexity of an application. That's where the Fused Location Provider comes in. Basically, it provides a high-level API that makes it easy to request location updates without needing to focus on the underlying details—location simplified.

Android's official Fused Location Provider is part of **Google Play Services** (<https://developer.android.com/google/play-services/index.html>), a collection of client libraries tightly integrated with Android. It's installed via the Google Play Store and updated

independently from the OS. It includes APIs for documents, games, in-app payments, maps, location, and other proprietary add-ons. However, there are some tradeoffs to using Google Play Services in your application.

Closed Source

Unlike the Android framework, Google Play Services is not open source. This provides less transparency and flexibility for developers. It also raises some questions regarding **security and privacy** (<http://commonsware.com/blog/2013/05/22/remember-google-play-services-proprietary.html>).

Application Size

Google Play Services increases the number of methods and overall size of your application. These effects can be somewhat mitigated by stripping unused methods with **ProGuard** (<http://developer.android.com/tools/help/proguard.html>) and enabling **multidex support** (<https://developer.android.com/tools/building/multidex.html>) but can still have implications for overall APK size and the 65K Dalvik Executable (DEX) method limit as well as for the complexity of your build process.

Device Compatibility

Since Google Play Services is only available on devices that support the Google Play ecosystem, it reduces the number of devices your app can run on. Your application will be incompatible with devices running a forked version of Android such as the **Kindle Fire** (<http://www.amazon.com/kindle-fire-hd-best-family-kids-tablet/dp/B00CU0NSCU>) or **Blackphone** (<https://blackphone.ch/>) as well as **most devices in China** (<http://techcrunch.com/2014/11/19/developers-in-china-can-now-make-money-via-google-play-apps-but-not-in-china/>).

This is where LOST comes in

LOST provides a pure open source implementation of the **FusedLocationProviderApi** (<https://developer.android.com/reference/com/google/android/gms/location/FusedLocationProviderApi.html>). Its only dependency is the **LocationManager** (<https://developer.android.com/reference/android/location/LocationManager.html>) so LOST can be used on any device that runs Android. It implements the same location APIs as Google Play Services so dropping it into an existing application is a snap.

Connecting to the LOST API Client

When using LOST, `GoogleApiClient` (<https://developer.android.com/reference/com/google/android/gms/common/api/GoogleApiClient.html>) is replaced by `LostApiClient` (<https://github.com/mapzen/LOST/blob/master/lost/src/main/java/com/mapzen/android/lost/api/LostApiClient.java>). Connecting to LOST is even easier since there are no `ConnectionCallbacks` (<https://developer.android.com/reference/com/google/android/gms/common/api/GoogleApiClient.ConnectionCallbacks.html>) or `OnConnectionFailedListener` (<https://developer.android.com/reference/com/google/android/gms/common/api/GoogleApiClient.OnConnectionFailedListener.html>) objects to manage.

```
LostApiClient lostApiClient = new LostApiClient.Builder(this).build();
lostApiClient.connect();
```

LOST instantly connects to the `LocationManager` (<https://developer.android.com/reference/android/location/LocationManager.html>) and can immediately retrieve that last known location or begin sending location updates.

Getting the Last Known Location

Once connected you can request the last known location. The actual logic to determine the best most recent location is based this classic **blog post by Reto Meier** (<http://android-developers.blogspot.com/2011/06/deep-dive-into-location.html>).

```
Location location = LocationServices.FusedLocationApi.getLastLocation();
if (location != null) {
    // Do stuff
}
```

Requesting Location Updates

LOST also provides the ability to request ongoing location updates. You can specify the update interval, minimum displacement, and priority. The priority determines which location providers will be activated.

```
LocationRequest request = LocationRequest.create()
    .setInterval(5000)
    .setSmallestDisplacement(10)
    .setPriority(LocationRequest.PRIORITY_LOW_POWER);

LocationListener listener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        // Do stuff
    }
};

LocationServices.FusedLocationApi.requestLocationUpdates(request, listener);
```

Currently location updates can only be requested with a **LocationListener** (<https://developer.android.com/reference/com/google/android/gms/location/LocationListener.html>) object. In the future we are planning to add location updates via a **PendingIntent** (<http://developer.android.com/reference/android/app/PendingIntent.html>) as well.

Mock Locations

With LOST you can mock not just individual locations but also entire routes. By loading a **GPX trace file** (<http://www.topografix.com/gpx.asp>) onto the device you can configure LOST to replay locations from the trace file including latitude, longitude, speed, and bearing.

Mocking a single location

To mock a single location with LOST you must first enable mock mode. Then you simply create a mock location object and pass it to the API.

```
Location mockLocation = new Location("mock");
mockLocation.setLatitude(40.7484);
mockLocation.setLongitude(-73.9857);
LocationServices.FusedLocationApi.setMockMode(true);
LocationServices.FusedLocationApi.setMockLocation(mockLocation);
```

The mock location object you set will be immediately returned to all registered listeners and will be returned the next time `getLastLocation()` is called.

Mocking an entire route

To mock an entire route you must first transfer a **GPX trace file** (<http://www.topografix.com/gpx.asp>) to the device using **adb** (<http://developer.android.com/tools/help/adb.html>). Sample GPX traces can be found on the **public GPS traces page** (<http://www.openstreetmap.org/traces>) for OpenStreetMap. Once the trace file is loaded on the device you can tell LOST to replay the locations in the trace at the requested update interval.

```
File file = new File(Environment.getExternalStorageDirectory(), "mock_track.gpx");
LocationServices.FusedLocationApi.setMockMode(true);
LocationServices.FusedLocationApi.setMockTrace(file);
```

For more in-depth examples, please refer to the **sample application** (<https://github.com/mapzen/LOST/tree/master/lost-sample>).

Feedback & Contributions

We originally created LOST for location services in the **Open by Mapzen** (<https://play.google.com/store/apps/details?id=com.mapzen.open>) application. We're excited to see how other developers will use LOST and are looking forward to evolving the project based on your feedback and contributions. If you're interested in using or contributing to LOST you can **fork us on GitHub** (<https://github.com/mapzen/lost>).

· 19 January 2015 ·



Chuck Greb

Chuck is an open source enthusiast, test-driven evangelist, and clean code connoisseur of all things mobile.

Mapzen's Code of Conduct Policy

Mapzen will only sponsor events with a strong code of conduct in place. Open source mapping is for everybody!



Every day at Mapzen we strive to build a healthier mapping ecosystem by ensuring it is diverse, sustainable, and accessible to all. One of the ways we do this is by sponsoring events in the geo and open source space. We know that these events are vital to our community, and many of us here have benefited tremendously from attending them. In 2014 alone, we were able to sponsor events like **State of the Map US** (<http://stateofthemap.us/2014/>), State of the Map 2014 (international), and **FOSS4G 2014** (<https://2014.foss4g.org/>). In addition, we **made a major donation to Code for America** (<http://www.codeforamerica.org/blog/2014/09/17/mapzen/>) to enable them to help governments open more geospatial data. We know that these sponsorships have real-world impact and have had a chance to see that first hand. Events like State of the Map and FOSS4G allow people in the community to make connections, learn about and present new technologies, and most importantly, meet in person.

Starting in 2015, Mapzen will only sponsor events with a strong code of conduct in place. We believe that codes of conduct are essential to ensuring that tech events are safe, diverse, and accessible places for all attendees, regardless of gender, gender identity and expression, sexual orientation, disability, physical appearance, body size, race, age, religion, or skill level. Sadly, the open source space isn't nearly as diverse as it could be. Something that is both a cause *and* effect of this is that sometimes tech events are hostile and unwelcoming to underrepresented groups, such as women or people with disabilities. We firmly believe that the mapping space should be as diverse as the world we're mapping, and we will only lend our name (and funds) to events that share that belief.

Codes of conduct aren't about limiting anyone's fun, or paying lip service to diversity, but instead they establish a clear, shared understanding of what behavior is unacceptable, and how violations will be addressed. They are an important signal to attendees that organizers care about everyone's experience, and will take seriously any reports of inappropriate behavior. Codes of conduct are certainly not the only thing needed to ensure a diverse and accessible event, but they are a step in the right direction. **PyCon**

(<https://us.pycon.org/2015/about/diversity/>) is a great example of a conference blazing the trail for inclusive tech events by going far beyond simply implementing a code of conduct.

We are following in the footsteps of companies we admire, **like Heroku**

(https://blog.heroku.com/archives/2013/12/11/code_of_conduct), by making this commitment public and by setting a clear understanding of what makes for a strong code of conduct. Our policy is strongly influenced by Heroku's, **which you can see here** (<https://www.heroku.com/policy/events>).

Mapzen will only sponsor events with a code of conduct in place that meets the following criteria:

- Provides an outline of behavior that is considered unacceptable
- Provides a system for reporting unacceptable behavior to event staff
- Makes clear the consequences for any behavior deemed unacceptable by the event staff

We encourage event organizers to look into the many resources available for writing strong codes of conduct. In particular, we are impressed by the work of **The Ada Initiative** (<https://adainitiative.org/>), and find their recommendations well thought out and approachable. We also encourage other companies to make similar commitments in public, so that event organizers know that this is something that is important to all of us.

If you are organizing an event and would like to know more about what it means to implement a strong code of conduct, **we recommend you start here** (<https://adainitiative.org/what-we-do/conference-policies/>). We also recommend that you take a look at these code of conduct templates from the **Geek Feminism Wiki** (http://geekfeminism.wikia.com/wiki/Conference_anti-harassment/Policy) (used by **PyCon** (<https://us.pycon.org/2015/about/code-of-conduct/>)) and **Stumptown Syndicate** (<http://citizencodeofconduct.org/>) (used by **Maptime** (<http://maptime.io/code-of-conduct/>)).

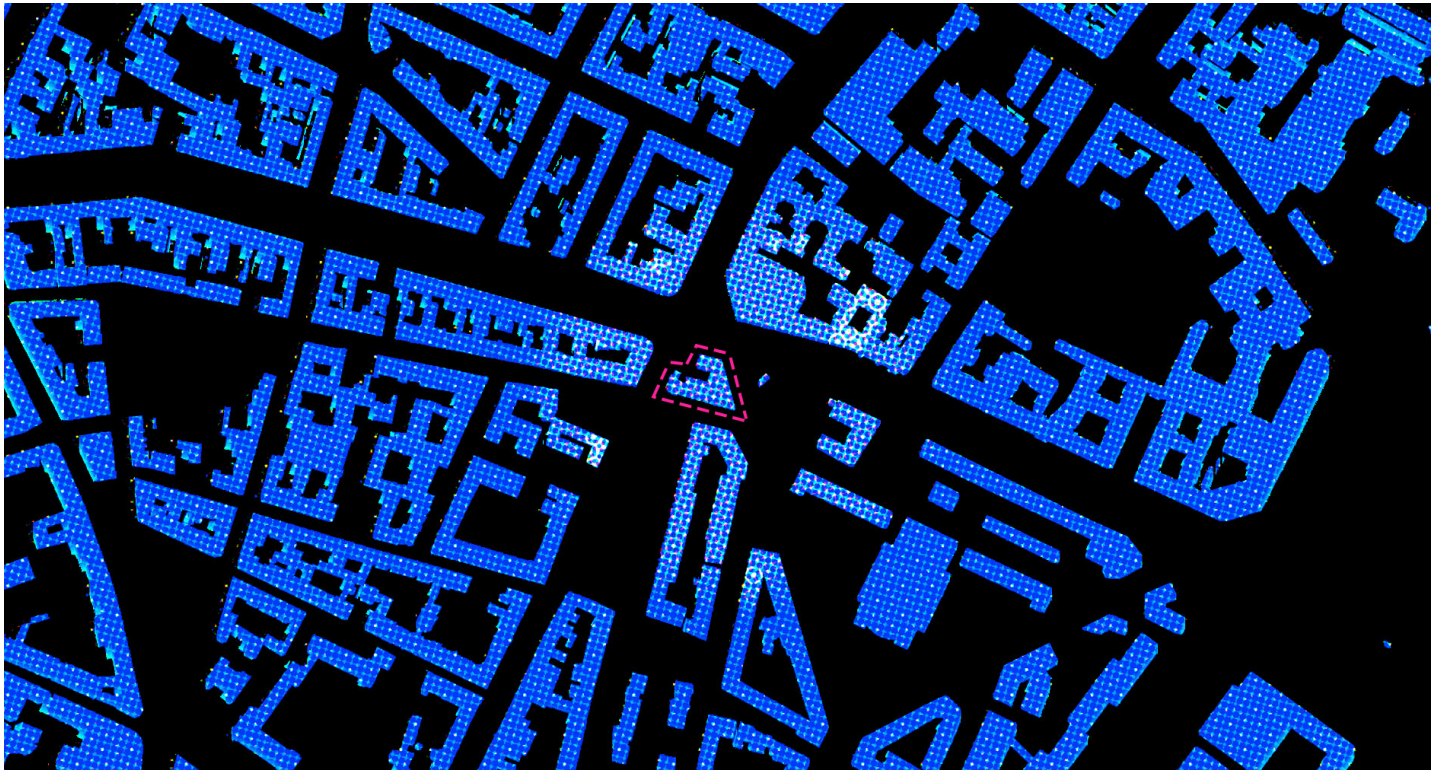
Kathleen Danielson is a Developer Advocate at Mapzen, and an advisor to The Ada Initiative.



Kathleen Danielson · 03 February 2015

© 2017 Mapzen

Hallo Berlin!



Mapzen may be based in NYC, but you can find us all over the world. Next week **Peter** (<http://twitter.com/insertcoffee>) and I will be joining **Kathleen** (<http://twitter.com/KathleenLD>) in Berlin, and we're hoping to meet up with you. Join us on Tuesday, February 10th at one of Kathleen's favorite Berlin hangouts, **Kaschk** (<http://kaschk.de>) (Linienstraße 40, 10119 Berlin). We'll be there from 7pm enjoying craft coffee, craft beer, Club Mate, and shuffleboard while we talk about maps, open source, and open data. Can't wait to see you there!

Bis bald!

· 05 February 2015 ·



Randy Meech

Billerica Memorial High School graduate. BA, MTS. Mapzen CEO 2013-present.

© 2017 Mapzen

It's a Marathon and a Sprint

You often hear the old adage repeated “It’s a marathon, not a sprint.” I believe this to be true for many things in life. Including open source projects. But sometimes sprints are good too.

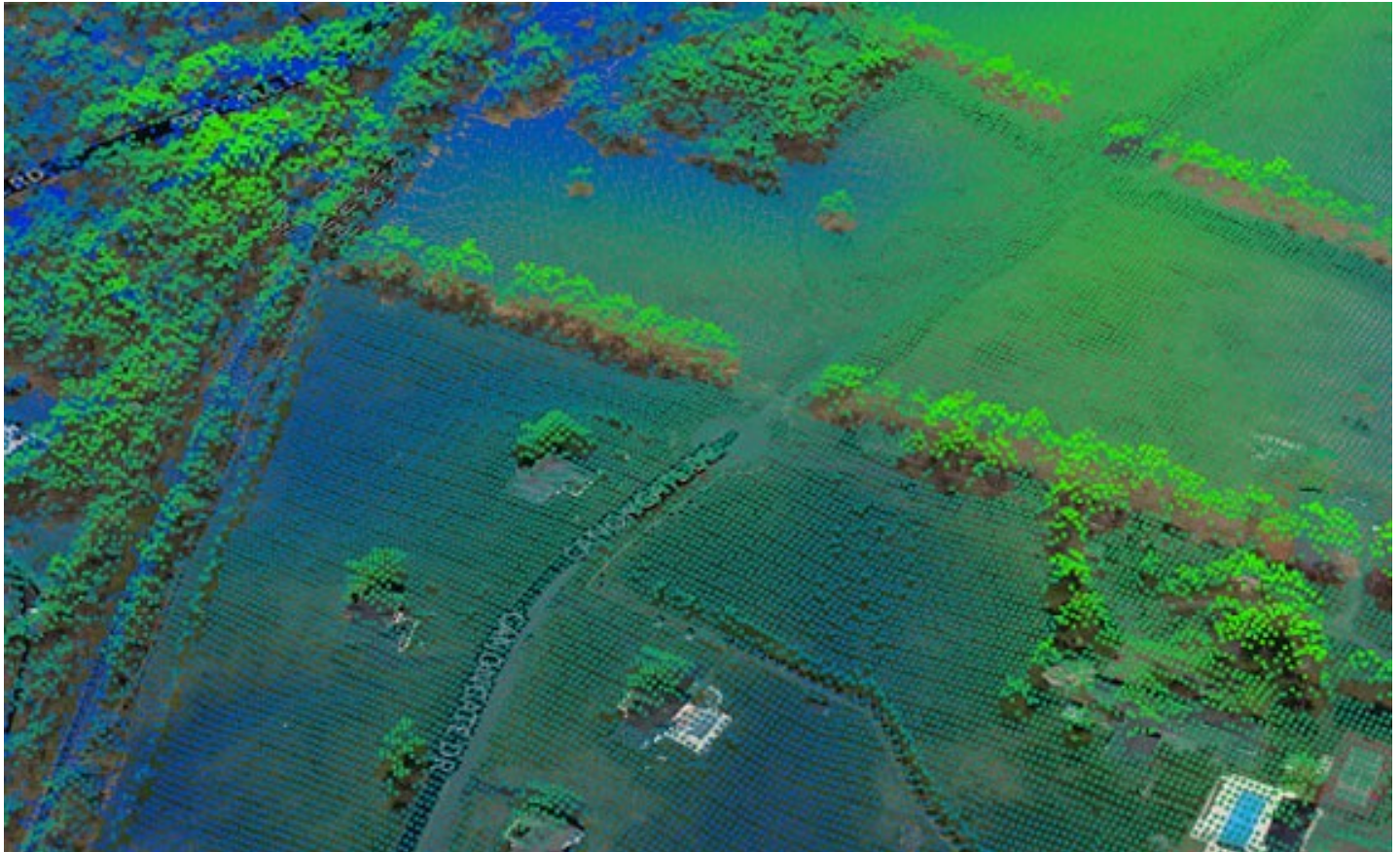


This week I attended the first day of the OSGeo **Philadelphia Code Sprint 2015** (http://wiki.osgeo.org/wiki/Philadelphia_Code_Sprint_2015) hosted by **Azavea** (<http://www.azavea.com/>). The goal of the week-long event is to bring together project members to make decisions and tackle larger geospatial problems in the open source geo community.

This year the OSGeo Code Sprint is a joint event with the Eclipse Foundation’s **LocationTech** (<https://www.locationtech.org/>) working group. There were representatives there leading teams working on a variety of projects including PostGIS, PDAL, GDAL, uDig, Cesium, and more. OSGeo and LocationTech are also teaming up to bring you **FOSS4G North America 2015** (<https://2015.foss4g-na.org/>) where several Mapzen folks will be presenting.

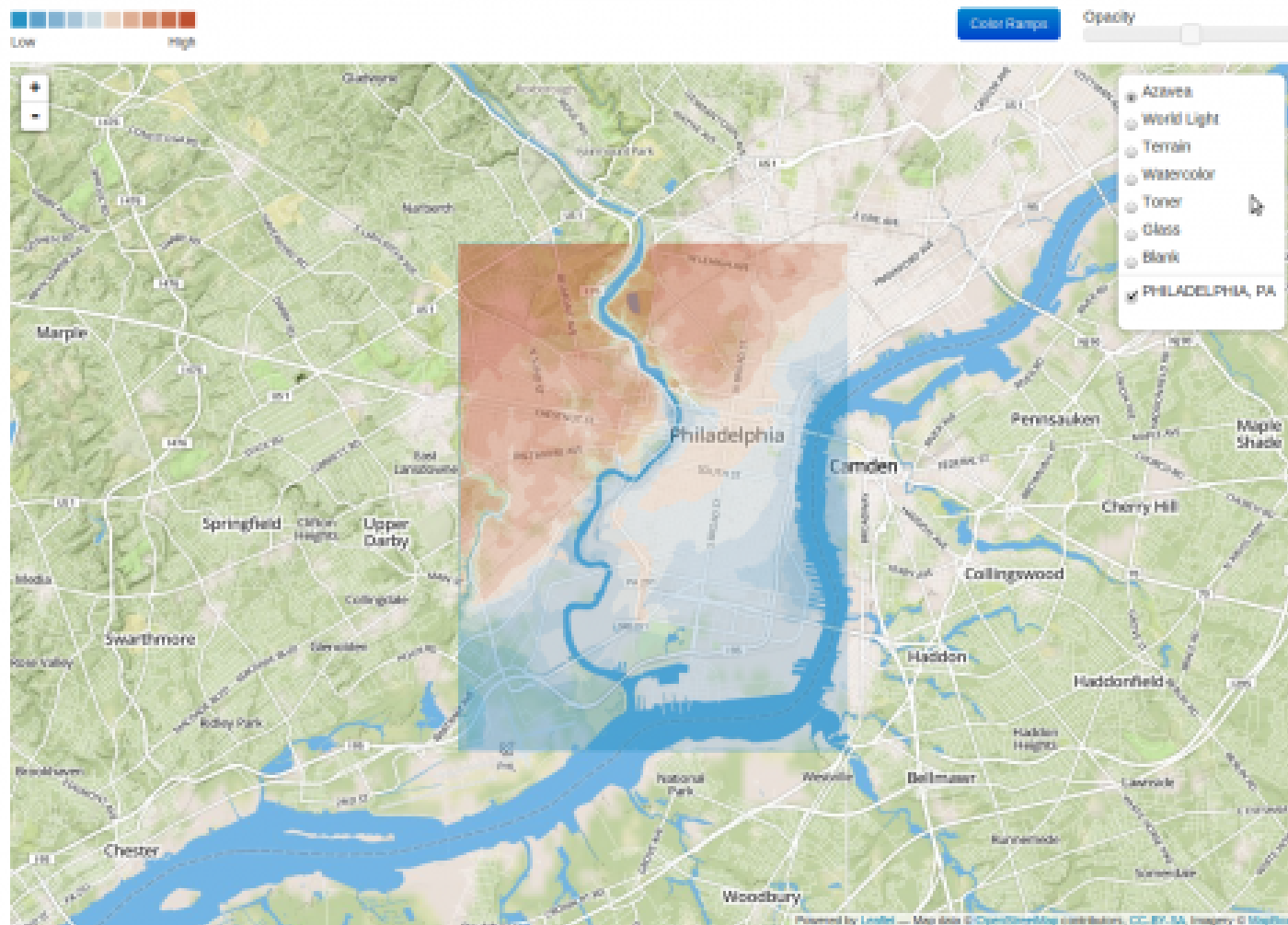
G3M

One project I found particularly interesting was the **G3M** (<https://github.com/glob3mobile/g3m>) data visualization tool created by **Diego Gomez Deck** (<https://twitter.com/diegogomezdeck>) of **Glob3 Mobile** (<http://www.glob3mobile.com/>). This mobile development framework supports complex visualizations with real-time data and full multi-touch support. I spent the morning helping Diego troubleshoot some device specific display issues on Android.



GeoTrellis

Later in the day I hooked up with the team working on **GeoTrellis** (<http://geotrellis.io/>), a high-performance raster processing tool. Thanks to a crash course in functional programming and some hand-holding from **Rob Emanuele** (<https://twitter.com/lossyrob>) I managed to make **my first open source contribution in Scala** (<https://github.com/geotrellis/geotrellis/pull/996>) to fix some intermittent test failures. I also helped resolve some **configuration issues** (https://groups.google.com/forum/#!topic/geotrellis-user/4R_QSEUZZ9A) when importing the project into **IntelliJ IDEA** (<https://www.jetbrains.com/idea/>).



The Code Sprint format provides an excellent forum for members of open source projects to collaborate on larger issues that may be difficult to resolve with distributed teams and asynchronous communication. Also the focused energy allows teams to push toward releases and other major milestones in a project. I hope to be able to attend more events like this in the future– possibly leading a team working on a Mapzen project like **LOST** (<https://mapzen.com/blog/lets-get-lost>). I've got my laptop, track pants, and sneakers all ready to go!

· 13 February 2015 ·



Chuck Greb

Chuck is an open source enthusiast, test-driven evangelist, and clean code connoisseur of all things mobile.

© 2017 Mapzen

Outreachy: Interview with HOT Intern Jessica Marlene Canepa

*This past October, Mapzen was **proud to announce** (</blog/foss-outreach-program-hot>) our sponsorship of one internship for the **Humanitarian OpenStreetMap Team (HOT)** (<http://hotosm.org/>) through **Outreachy** (previously known as the **Outreach Program for Women**) (<https://www.gnome.org/outreachy/>). Outreachy provides funding for FOSS internships for women (cis and trans), trans men, genderqueer people, and all participants of the **Ascend Project** (<http://ascendproject.org/>) regardless of gender. We were delighted to learn that the GNOME Foundation had heard about our sponsorship and decided to match the funds we provided, allowing for a second intern for HOT. We wanted to give you an update on the program's progress, so we took the opportunity to ask **Jessica Marlene Canepa** (<http://jmarlena.github.io/>), one of the interns, some questions about her experiences with the program.*

*This is the first blog post in a series of interviews with Outreachy interns and mentors. Read our interviews with intern **Nitika Agarwal** (</blog/outreachy-interview-nitika>) and mentor **Kate Chapman** (</blog/outreachy-interview-kate>).*

How did you learn about the Humanitarian OSM Team? What attracted you to working with them?

I learned about the Humanitarian OpenStreetMap Team (HOT) from the Outreach Project for Women (OPW) application website. I heard about OPW, which is undergoing a name change to Outreachy, from volunteering at the **Open Source Bridge** (<http://opensourcebridge.org/>) conference in my hometown of Portland, Oregon. I have heard of OpenStreetMap (OSM) but not the Humanitarian OSM Team (HOT) before applying for the OPW internship. I was excited by HOT's global impact, capacity building focus, and collaborative approach in using open source tools. As an undergrad I studied International Affairs and Russian so HOT's international mission and community is very appealing to me. I have always loved surrounding myself with maps and learning from others around the world.

Tell us about the OPW program. What kind of structure and support is provided for interns?

The OPW program offers mentorship for passionate newcomers in the form of a remote internship. OPW interns span the globe. They have a great internship coordinating team and online community that includes office hours for career related questions, an IRC (Instant Relay

Chat) room, LinkedIn group and other online places for communicating with current and former interns and mentors.

To apply for the internship OPW applicants must have already reached out to the project's mentors and have started to make a contribution. This helps applicants get to know the organization, mentor(s) and project details before officially starting.

OPW mentors and interns set up the expectations of the internship at the beginning, including the frequency of their weekly communication and check-ins. They're expected to check-in at least once a week but often have other built-in ways they communicate throughout the week in the form of work logs, IRC, voice chat, and email for updates and troubleshooting help.

What have you been working on since the start of your internship?

I have been working on a documentation project to help others get started in participating in HOT with updated guides and with helping with GitHub issues for the learnosm website.

What has surprised you the most in the course of your internship so far?

I have been most surprised by the open source process of collaborative decision making and the need for clear communication. I didn't expect collaboration to take so much time and didn't realize how critical clear communication is to the process especially with an international community of contributors.

What are your plans for after the internship is over?

After the internship I plan to keep learning by finding other open source projects (preferably map or geo related) to help with in order to hone my skills. I am really interested in building the skills of a mapping web applications developer familiar with open source tools.

What advice would you give someone considering applying for the OPW internship?

My advice to someone considering an OPW internship is to find a project that really intrigues you or an organization's mission that you feel passionate about. I originally intended to apply for multiple organizations as a part of my OPW application but quickly found HOT's mission most motivating and settled on just applying for HOT's projects given my limited time. As a part of my application I got stuck on installing PostgreSQL to run the HOT Tasking Manager website locally on my computer and I spent several days troubleshooting this. I realized that I wouldn't have had the same motivation for other projects I considered and that I could probably learn the most from being invested in the organization's mission.

*Outreachy is getting ready for their summer internship and Mapzen is planning to sponsor another intern. If your company would like to sponsor an internship for an open source project, **get in touch with Outreachy now** (<https://www.gnome.org/outreachy/#apply>). Applications for the internships will open March 3, 2015.*

· 20 February 2015 ·



Kathleen Danielson

Berlin-based Developer Advocate. OSM Foundation board member. Mental health, feminism, FOSS, communities, cats, craft beer.

© 2017 Mapzen

Outreachy: Interview with HOT intern Nitika Agarwal



Image cropped from **GNOME Project Outreachy** page (<https://www.gnome.org/outreachy/>). CC-BY (<http://creativecommons.org/licenses/by/3.0/>)

This past October, Mapzen was **proud to announce** (</blog/foss-outreach-program-hot>) our sponsorship of one internship for the **Humanitarian OpenStreetMap Team (HOT)** (<http://hotosm.org/>) through **Outreachy** (previously known as the **Outreach Program for Women**) (<https://www.gnome.org/outreachy/>). Outreachy provides funding for FOSS internships for women (cis and trans), trans men, genderqueer people, and all participants of the **Ascend Project** (<http://ascendproject.org/>) regardless of gender. We were delighted to learn that the GNOME Foundation had heard about our sponsorship and decided to match the funds we provided, allowing for a second intern for HOT. We wanted to give you an update on the program's progress, so we took the opportunity to ask **Nitika Agarwal** (<http://nitika-opw2014.blogspot.in/>), one of the interns, some questions about her experiences with the program.

This is the second blog post in a series of interviews with Outreachy interns and mentors. Read our interviews with intern **Jessica Marlene Canepa** (</blog/outreachy-interview-jessica>) and mentor **Kate Chapman** (</blog/outreachy-interview-kate>).

How did you learn about the Humanitarian OSM Team? What attracted you to working with them?

The Humanitarian OpenStreetMap Team applies the principles of open source and open data sharing for humanitarian response and economic development. HOT works both remotely and physically in countries to assist the collection of geographic data, usage of that information and

training others in OpenStreetMap.

OpenStreetMap is a project to create a free and open map of the entire world, built entirely by volunteers surveying with GPS, digitizing aerial imagery, and collecting and liberating existing public sources of geographic data.

OSM Tasking Manager (<http://tasks.hotosm.org/>) is a mapping tool designed and built for the Humanitarian OSM Team collaborative mapping. The purpose of the tool is to divide up a mapping job into smaller tasks that can be completed rapidly. It shows which areas need to be mapped and which areas need the mapping validated. This approach facilitates the distribution of tasks to the various mappers in a context of emergency. It also permits to control the progress and the homogeneity of the work done (ie. Elements to cover, specific tags to use, etc.).

While researching the different project ideas proposed by the participating organisations in this OPW program, I found the project “Improving the OSM Tasking Manager v2 Homepage” along with two other project ideas very appealing for me, to contribute to them as a prospective participant of the OPW internship. Also, I found the organisation quite appealing, because of the work done by the community members and the possibility of contributing to the real open source project, and working with the experienced and motivated mentors. Then, I started working on submitting the patch and preparing the project proposal application.

Tell us about the OPW program. What kind of structure and support is provided for interns?

Outreach Program for Women, OPW is a free and Open Source program organised by the Gnome Foundation that motivates women to get involved in the Open Source Software by offering internships (of around 13 weeks) twice a year. The best part of the OPW program is that it allows newcomers to work on open source projects.

OPW is an excellent program through which you can get in touch with experienced and motivated mentors who are always willing to help you with any sort of issue throughout the internship period and resolving your queries. And you receive a decent amount of stipend for your hard-work in return!

Research the project ideas: First research options on the project ideas page proposed by the developers of the participating organisations if you are interested in to work on and have the desired skills to work on that project idea.

Discussion with the Project mentors: Communicate with the project mentor and other developers of the organisation on their IRC channel to learn more about the project idea.

Submit an initial patch: After getting enough of an idea of the project and the organisation, submit a patch by resolving some already present bugs which will show your skills and interest in the organisation to the coordinators and developers. Don't hesitate to ask for help from the developers on the IRC channel.

Project Proposal: After getting enough of an idea of the project, start working on the project proposal during the application period. Include your project abstract, a detailed solution proposed, tentative timeline to complete the project tasks, information about yourself, your academic background, contact details and why you think you are the most suitable to work on this particular project.

Submit the application: Now you need to submit the project application on the Gnome website before the specified deadline.

[Ed. Note: Instructions for Summer 2015 applicants will be published in the coming weeks]

What have you been working on since the start of your internship?

I'm working on the HOT Tasking Manager v2 homepage with the mentors Pierre Giraud and Kate Chapman. The project involves improving the homepage of the HOT tasking manager v2, since the current design makes the other projects (jobs) such that the last / most urgent ones hide all the rest. The project was split into modules: Feature request analysis, Frontend Design (User Interface), Community feedback and discussions, Backend Design (mostly for the database), Implementation and Testing. Currently I'm working on the implementation part of the project.

What are your plans for after the internship is over?

After the end of the OPW internship, my plan is to contribute to the OpenStreetMap organisation as an active member of the community. I'll be around to help others as best as I can and contribute to the OSM Tasking manager project to extend the functionality or work on other project ideas. I'm also planning to get involved with other open source organisations and contribute to them.

What advice would you give someone considering applying for the OPW internship?

For the candidates looking to apply for the OPW Internship, I would like to suggest some things to keep in mind while working on the OPW application:

- **Be Honest:** Be honest about your experiences and the skills you have to work on the project, identifying the open source applications that you are currently using or have used

in the past. Don't over-inflate what you have done but don't gloss over it either.

- **Be Curious and Be Yourself:** As you work on researching the different project ideas proposed by the open source organisations for the OPW Internship and then contacting the mentor to introduce yourself and then discuss the project idea, be yourself. Lurking on the IRC channel of the organisation and sending mails to the org's mailing list are the best means for contact to seek help and guidance from the developers.
- **Communicate your Status:** You should communicate about the progress of the project work on a regular basis to the mentors and the community developers through emails, weekly meetings on IRC and blog posts. Commit your code frequently. This will really help you to get proper guidance and feedback from the mentors, also the mentor will be updated on what you are working on.
- **Plan:** Nothing can be achieved without planning. So you need to plan out the tentative timeline for the completion of the project's tasks and mention it in the application.
- **Be Patient:** Getting started is the hardest part in any new endeavour. So, you need to be patient until you have the sound grasp of the developer tools and understanding of the underlying APIs which will lay the foundation of your contributions.
- **Complete the Patch Submission Requirement:** Your application will not be considered without the patch submission. It is possible to submit a patch on short notice, but submit a patch either to fix some bug or develop new feature. The patch submission will show that you have the desired skills and experience with common tools used by the organisation.
- **Have Fun:** There will be times of both frustration and great pride. Allow yourself to experience both and have fun while working.

Making a difference in open source software requires leadership, hard work and problem-solving. OPW, in my opinion, has been a great success due to the dedication from the leaders and the GNOME community. The OPW program has been embraced by many FOSS communities.

Disclaimer: Please note that the above answers are my own thoughts and suggestions and don't hold me or the community responsible for any information mentioned above.

*Outreachy is getting ready for their summer internship and Mapzen is planning to sponsor another intern. If your company would like to sponsor an internship for an open source project, **get in touch with Outreachy now** (<https://www.gnome.org/outreachy/#apply>). Applications for the internships will open March, 3, 2015.*

· 23 February 2015 ·



Kathleen Danielson

Berlin-based Developer Advocate. OSM Foundation board member. Mental health, feminism, FOSS, communities, cats, craft beer.

© 2017 Mapzen

Outreachy: Interview with HOT Executive Director Kate Chapman



Kate Chapman, Executive Director of HOT

This past October, Mapzen was **proud to announce** ([/blog/foss-outreach-program-hot](https://blog/foss-outreach-program-hot)) our sponsorship of one internship for the **Humanitarian OpenStreetMap Team (HOT)** (<http://hotosm.org/>) through **Outreachy** (previously known as the **Outreach Program for Women**) (<https://www.gnome.org/outreachy/>). Outreachy provides funding for FOSS internships for women (cis and trans), trans men, genderqueer people, and all participants of the **Ascend Project** (<http://ascendproject.org/>) regardless of gender. We were delighted to learn that the GNOME Foundation had heard about our sponsorship and decided to match the funds we provided, allowing

for a second intern for HOT. We wanted to give you an update on the program's progress, so we took the opportunity to ask HOT Executive Director Kate Chapman, one of the mentors, some questions about her experiences with the program.

*This is the third and final blog post in a series of interviews with Outreachy interns and mentors. Read our interviews with interns **Jessica Marlene Canepa** (</blog/outreachy-interview-jessica>) and **Nitika Agarwal** (</blog/outreachy-interview-nitika>).*

Why did you decide to get involved in the OPW as a mentor?

I had been looking to get involved in OPW for a while; I have strived to help women get involved in open-source for a while. I previously had an intern join me in Jakarta and also was a mentor for Google Summer of Code one year.

How was the OPW different from other experiences you've had finding interns in the past?

Since I'd work on Google Summer of Code it wasn't too different from that. I do think since it didn't require specifically programming projects we were able to get a greater variety of people.

What kind of difference have OPW interns made for HOT? Would you recommend other FOSS projects get involved in the OPW?

Having someone look at something with fresh eyes is great. I've been involved in HOT for 5 years, it is easy to forget what it is like to be new. I certainly would recommend other organization to get involved, especially if they are concerned about diversity.

OPW interns are paid. Does that make a difference in the background of applicants and eventual interns you're able to employ?

I'm not sure, we haven't really had many unpaid interns before. The few interns we've had were volunteers in specific missions in different parts of the world rather than where they already lived. So in those cases we were paying living expenses.

What is the level of commitment required from you as a mentor? What advice would you give someone thinking of being a mentor?

I spend roughly two hours a week per intern (I support two). It has been difficult this round to always be available because I've traveled a lot. My preference would be to mentor during periods where I have less travel. As far as advice, I think expectations are important. This is probably true for most of life, if you set expectations then people know what is required of them. So in addition to the OPW requirements we also wrote our own expectation documents.

Would you get involved in the OPW again? What would be your goals for future sessions?

I would certainly get involved again. I think in the future I'd work a bit harder to set up clear milestones. Everyone has done a great job, but I think we could have made the steps a little more clear.

What would you say to organizations or companies considering hosting or sponsoring interns?

OPW is a great program. It is important to have enthusiastic mentors. The most ideal situation to me is that a company would give mentors work time to assist.

*Outreachy is getting ready for their summer internship and Mapzen is planning to sponsor another intern. If your company would like to sponsor an internship for an open source project, **get in touch with Outreachy now** (<https://www.gnome.org/outreachy/#apply>). Applications for the internships will open March, 3, 2015.*

· 24 February 2015 ·



Kathleen Danielson

Berlin-based Developer Advocate. OSM Foundation board member. Mental health, feminism, FOSS, communities, cats, craft beer.

© 2017 Mapzen

Barcelona Bound!



Alyssa and I will be in Barcelona next week for **Mobile World Congress** (<http://www.mobileworldcongress.com/>). We're looking forward to talking with people across the mobile industry to see what new projects everyone is working on, and sharing our work from Mapzen.

I'll be taking part in the panel "The Next Generation Mapping Platform will be Open Source," with some of the brightest minds in open geo. I'm excited for a great forward-looking conversation with CartoDB's **Javier de la Torre** (<https://twitter.com/jatorre>), Mapbox's **Eric Gundersen** (<https://twitter.com/ericg>), and **Tyler Bell** (<https://twitter.com/twbell>) from Factual. Thanks in particular to CartoDB for pulling this panel together.

Catch the panel on Wednesday, March 4 at 15:30 CET in the presentation area of the Spanish Pavilion (Congress Square CS60).

We'll also be at a geobeers event on Wednesday night from 20:00 CET at the itnig space: Carrer Àlaba 61, 5-2 Barcelona. Stop by if you're free or drop a line if you want to meet up!

· 26 February 2015 ·



Randy Meech

Billerica Memorial High School graduate. BA, MTS. Mapzen CEO 2013-present.

© 2017 Mapzen

Mapzen's Support for Code for America and OpenAddresses

Mapzen (<http://mapzen.com>) has been supporting Code for America's work on geographic open data for the past half-year.

"Land parcels are the atomic unit of government's view of the physical world."

We began our collaboration with Mapzen at the idea of research for an open parcels data standard. Land parcels are important, they are the atomic unit of government's view of the physical world. It's also a confusing term, as we learned talking with experts like Nancy Von Meyer of **Fairview Industries** (<http://www.fairview-industries.com>).

There are ownership parcels, rights parcels, assessment parcels, usage & zoning parcels tracked at many different levels. For Code for America's interest in economic development, real estate, or tax parcels looked like they could be very valuable. Parcel datasets might include information about ownership, value, tax rate, and relationship to municipal zoning codes that can help established and new businesses choose locations. Easy-to-understand data on parcels would be useful to entrepreneurs, residents, and civic startups like **Open Counter** (<http://opencounter.us/>) and their **Zoning Check product** (<http://zoningcheck.us/>).

The problem is that open standards are a notoriously tough nut to crack, and we did not want to end up with yet another unused standard in a world with many existing standards. This problem is shown well in the below xkcd cartoon:

HOW STANDARDS PROLIFERATE: (SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)

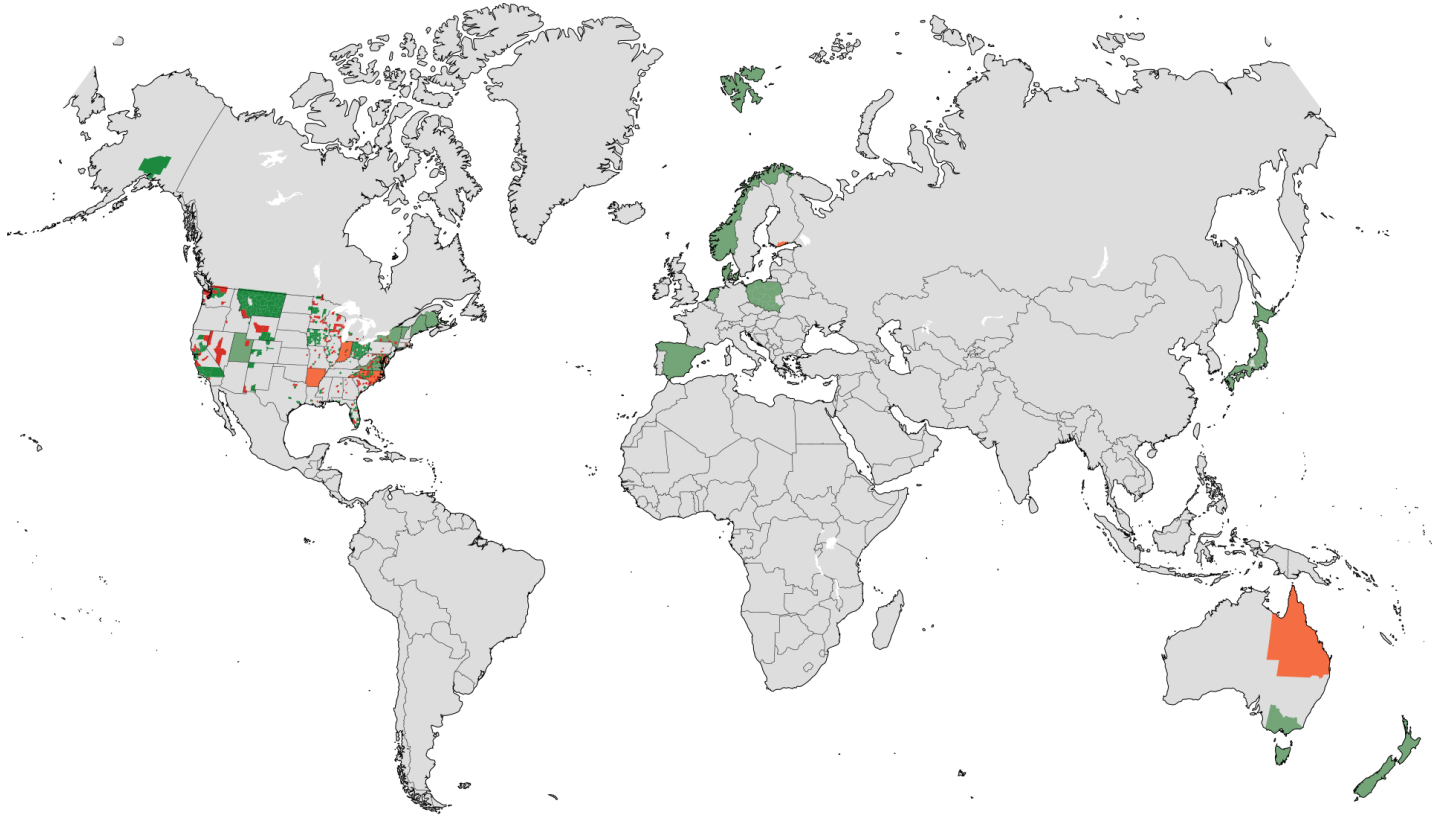


(<http://xkcd.com/927/>)

XKCD, No. 927 (<http://xkcd.com/927/>)

So, we chose to look at existing data via an ongoing project, **OpenAddresses** (<http://openaddresses.io/>), to decide what we could learn from data that's already open and available to use. Addressing and parcels are not the same thing, but in a country of single-family homes they overlap significantly enough to be worthwhile. Land parcels are a common source of open data used in the year-old OpenAddresses project, and addresses are a crucial component of any full-stack mapping service. Georeferenced address services come up every year in Code for America's fellowship program, most recently in **Team Lexington's 2014 Streetscope project** (<https://github.com/codeforamerica/streetscope/>).

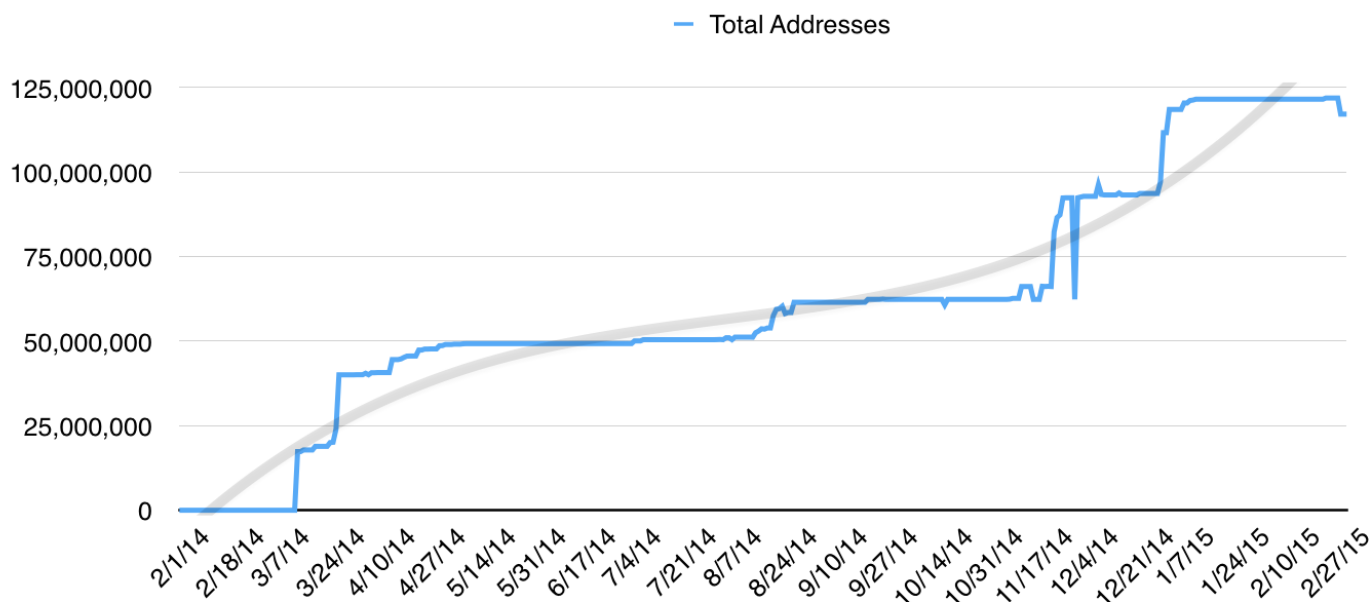
OpenAddresses (<http://openaddresses.io/>), started by Ian Dees in spring 2014, is a worldwide searchable database of roughly 100 million address points. It collects publicly available, open land parcel and address data, and produces a single output file that covers the world.



(<http://data.openaddresses.io/>)

Based on experience from the early days of **OpenStreetMap** (<http://openstreetmap.org/>), we believe that providing a visible peek into the state of the project will spur contributions, so we've been working to build and improve **data.openaddresses.io** (<http://data.openaddresses.io/>), a service that makes it much easier to see the result of adding data to the OpenAddresses database. It does this by performing continuous integration on the OpenAddresses database, automatically building it and visually showing the changes every two days.

We've been running this process since the beginning of November, 2014. It's helped OpenAddresses to identify and fix issues in many data sources:



(<http://www.codeforamerica.org/blog/wp-content/uploads/2015/02/openaddresses-totals.png>)

We've created a positive feedback mechanism to accelerate OpenAddress's growth, and a place to collaborate with other members of the open data community like Tom Lee, Nick Ingalls, Nelson Minar, and OpenAddresses founder Ian Dees. Our close relationship with Mapzen allowed us to identify OpenAddresses as a place where we could make a difference in the quality and quantity of foundational open data for the civic ecosystem.

What's next? If you're a data owner at the city or county level, it's a great time to contribute to Open Addresses and help build a valuable piece of open geographic data infrastructure. To get started and add a source, check out <http://openaddresses.io/contribute/> (<http://openaddresses.io/contribute/>).

Michal Migurski is CTO at Code for America. He **originally posted this** (<http://www.codeforamerica.org/blog/2015/03/03/mapzen-and-our-work-on-geographic-open-data/>) on the Code for America blog.



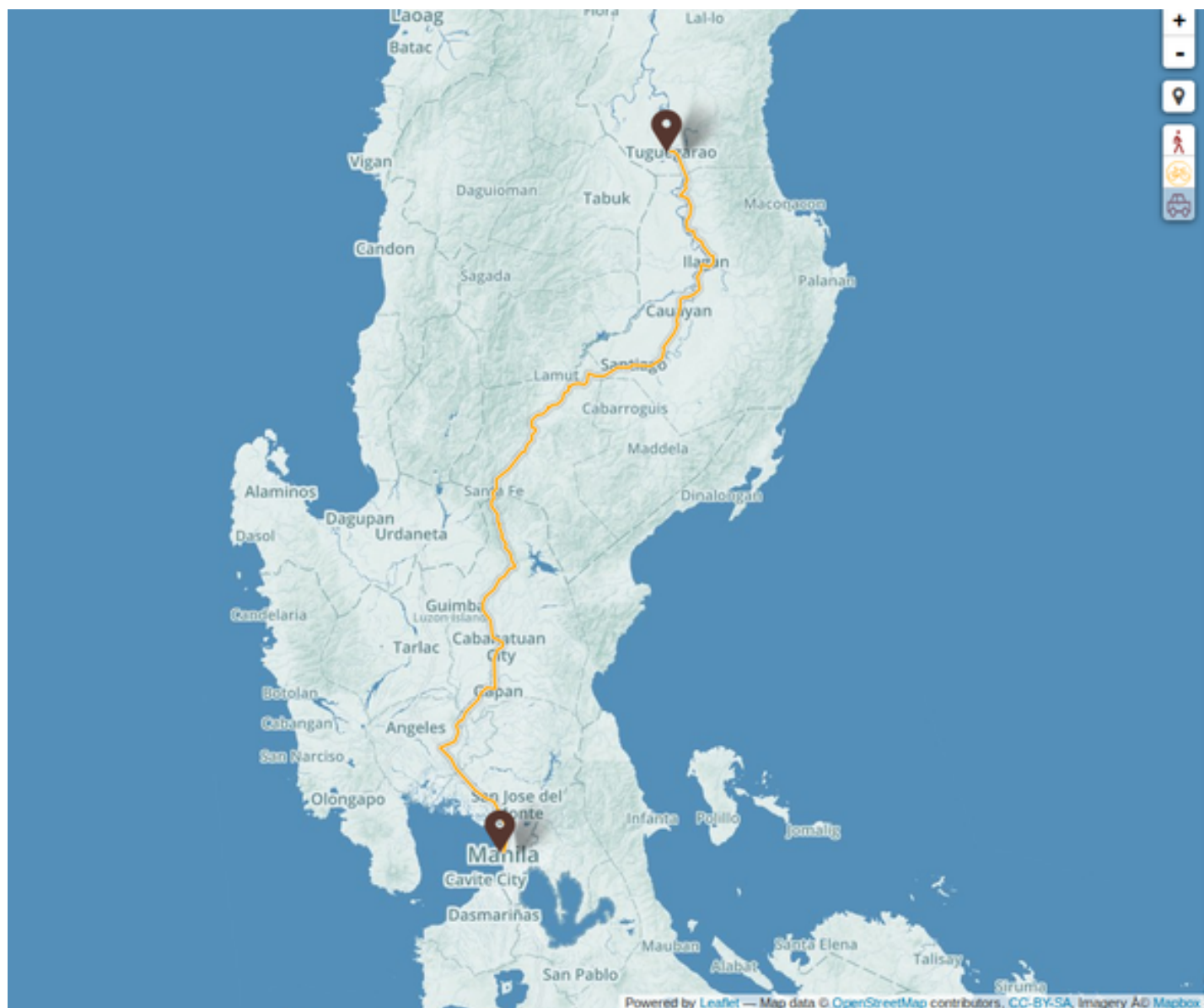
Michal Migurski · 03 March 2015

© 2017 Mapzen

Valhalla Open Source Routing

routing (/tag/routing)

Routing is a wily beast and many valiant efforts have been made to slay it, with limited success. However, we have brought a mighty new Mjölnir-like weapon to bear on this beast: Open Source!



After much intense battle development we are proud to take the wraps off of Valhalla. Valhalla is open source routing software using open source data (primarily Open Street Map), with a very liberal license. This should allow for transparency in development, encourage contribution and

community input, and foster use in other projects. The name is inspired by key features of the routing engine: the core route engine is called THOR (Tiled, Hierarchical Open Routing), generation of trip information for the path is called ODIN (Open Directions and Improved Narrative) and the service component is called TYR (Take Your Route). Valhalla seemed like a fitting organization name – previous efforts may have died but we all get to fight on in the great hall that is Open Source.

Valhalla developers know routing. We have worked together as a team for over a decade and have complementary skills that allow efficient software development. Our experience includes both commercial and OSM routing. We are well versed in scaling routing services to handle millions of requests per day. Performance and scalability is important, but our primary focus will be creating high quality routes, guidance, and directions. Existing open-source routing engines derive from academic research, resulting in fast routing algorithms over large graphs/networks. There are many properties of road networks that must be considered to produce quality routes and guidance/narrative descriptions. Rather than wrap these decisions into a baked costing model we choose to add most attribution to the graph data and allow dynamic, run-time costing. This should allow others to contribute and apply costing models and also allow flexible use of alternate costing to produce routes with different characteristics.

There are other key features that we hope distinguish Valhalla from other systems and encourage developers to build systems around the service and contribute back to the project. These features include:

Multi-modal and Time-Based Routing

While the initial development phase of Valhalla will focus on single mode trips, we want to quickly move to support mixing auto, pedestrian, bike and public transportation in the same route. Support for public transit requires time and schedule dependent routing and must support tracking time along a path and can potentially support setting a time by which one must arrive at a location. From day one the design of Valhalla has been influenced by multi-modal considerations.

Tiled, Hierarchical Data

Lets face it, building routing data sets from OSM is not easy. We wondered why routing data couldn't be treated like vector map data - use a tiled data structure to allow easy downloading and updating of regions. Graph (the route data structure) tiles can be downloaded for use by client-side routing applications or by hosted services that don't want to go through the pain of data creation. A structured graph hierarchy (e.g., highways, arterials, local, transit) along with

shortcut edges will ensure high performance. **THOR** (<https://github.com/valhalla/thor>) should allow for smaller memory footprints on memory constrained devices and provide a means for regional extracts and partial updates.

Take Your Route (TYR)

Tyr (<https://github.com/valhalla/tyr>) signifies another fitting theme both from Norse mythology and from our open approach. Initially, Tyr will be our routing service where users can generate routes for mobile or web use. We plan to build methods to download tiled route data to allow unconnected, client-side features like off-line routing where users can “take your route” on the road or download graph tiles for a region and be able to use their device in places they might not have connectivity. Features like client-side “return to route” and off-line routing are possible.

Flexibility and Extensibility

We want to encourage others to contribute their expertise and local knowledge to routing and guidance/narrative. What makes a good route in one country/region may not hold true in another country. Having the ability to create dynamic and extensible “plug-in” code to perform costing/weighting may encourage others to use and extend Valhalla. Dynamic costing will also help create alternate route paths (at run time - without generating different data sets) and allow new costing methods for specialized use cases: truck routing, green/eco routing, and perhaps least cost routing. Within narrative and guidance generation software we want to provide means of adding custom narrative phrases and perhaps other means of extending or adding custom plug-ins to tailor the output to a user’s need.

Open Directions and Improved Narrative (ODIN)

A quality route result is more than just a path shape and long list of road names with simple turns and dreaded “continue” instructions. **ODIN** (<https://github.com/valhalla/odin>) will be responsible for transforming path information into guidance and narrative directions that are easy to understand, useful, and assist users during their trip. Exit information and directional information on highways will help remove ambiguity at key decision points along the route. For example: - Take exit **51B** on the **right** onto **I 81 North** toward **I 78/Hazleton/Allentown**

ODIN collapses maneuvers using common base street names and simplifies transitions at complex intersections. Landmarks and other related information are also planned. Guidance and route explication must also be able to be tailored to different languages and potential uses - so extensibility and contributions from others are key.

What is Next?

We have big plans! Look for an announcement of a hosted service in the near future. Transit support should follow soon afterwards. Also look for regular blog posts that provide more technical details on the various components of Valhalla. In the meantime follow our progress on Github under the Valhalla organization: **Valhalla** (<https://github.com/valhalla>).

Our Team of Valiant Warriors

Duane Gearhart (<http://twitter.com/DuaneGearhart>) Guidance, narrative, quality expert.

Greg Knisely (<http://twitter.com/heyknisely>) OSM and transit data expert.

Kevin Kreiser (<http://twitter.com/kevinkreiser>) C++, systems design, open source enthusiast and computational geometry nerd.

David Nesbitt (<http://twitter.com/dnesbitt61>) Routing data and algorithm design.

· 20 March 2015 ·



David Nesbitt

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.

© 2017 Mapzen

Why Tiles

routing (/tag/routing)

Routing Tiles – the Who, What, When, Where, How and Why?

A bit of the Who, When and How

We had just started our new endeavor at Mapzen and were kicking around fundamental ideas of what our system should look like. Here's a bit of a transcript of how we got here:

17/11/14 01:27 EST **Kevin:** *wakes up in a cold sweat and begins to fumble around for his phone*

17/11/14 01:28 EST **Kevin:** *violently stubs his toe on the sharpest lego he's ever witnessed*

17/11/14 01:30 EST **Kevin:** *cursing and slightly bloody, he manages to ring Dave*

17/11/14 01:32 EST **Dave:** Uhnng...

17/11/14 01:32 EST **Kevin:** Dave! What if we route on tiled data!?

17/11/14 01:33 EST **Dave:** Um... yeah... you've been working in the mapping world for far too long

17/11/14 01:33 EST **Dave:** *throws his phone to the furthest corner of the room*

17/11/14 01:34 EST **Kevin:** Dave??

17/11/14 01:35 EST **Kevin:** *begins pacing*

17/11/14 04:17 EST **Kevin:** *still pacing*

17/11/14 07:01 EST **Kevin:** *rings dave once again*

17/11/14 07:02 EST **Kevin:** But Dave... We can have levels of detail, regional extracts, offline mobile routing... and we can open source it all

17/11/14 07:03 EST **Dave:** Hmm... Tiled. Hierarchical. Open. Routing. We could call it THOR. That's a pretty sweet acronym! That will make a great project name. But yeah... I'm not sure it'll work.

Ok, so the above is slightly embellished (everyone knows Dave stopped answering his phone after that one weekend call about traffic on the NJ Turnpike), and let's just say Kevin's proposal was met with some degree of skepticism. But after reasoned consideration the team decided that tiled, hierarchical routing data had merits and was worth pursuing.

Several months later, we have found that a tiled routing graph is possible and have begun exploring some of the benefits we'd hoped it would provide us and especially you!

This GIF gives a conceptual overview of the tiles used in a resultant graph traversal and how the level of detail needed may change depending on the connectivity of the road network in a given region.



(<https://mapzen-assets.s3.amazonaws.com/images/why-tiles-post/whytiles.gif>)

Show me the What, Where and Why!

So you can find all of our software at our **Valhalla** (<https://github.com/valhalla>) github organization. There's a lot of software there but most of it is modern c++ with a little bit of python for prototypes, a bit of ruby for chef style deployment and a bit of javascript for demo'ing

some of the functionality.

Onto the Why!

We hope to enjoy some key benefits of a tile based approach to routing, they are as follows:

- **Reduced memory requirements:** a connected graph can take up a lot of space in memory. By cutting the graph into a tiled structure you more easily impose limits on how much of the graph resides in memory at any one time. This could enable on device routing capabilities on even the most meager of hardware which could improve access to those in developing regions.
- **Cacheability:** imagine a multi-level cache where an S3 bucket has a setting of constantly updated route tiles with proper http cache headers etc. Follow that up with a client side disk cache used to back a memory cache that the routing algorithm actually uses in graph traversal computations.
- **Updateability:** updates to the graph could be highly parallelized simply because the graph is already broken up into a tiled structure. Faster turn around times on edits would be spectacular.
- **Regional Extracts and Off-line Routing:** you're headed to Switzerland for vacation and you don't want to pony up the Francs to get a SIM card? Before you go, or while you're on wifi, download an extract of the region. Afterall it's only 100mb or so.
- **Return to Route:** most phone based navigation applications require contacting a server when a user deviates from the specified route path, even for common cases like stopping for fuel or food. With graph tiles downloaded along the route path, the navigation application can find a path back to the route without an expensive call to the routing server.
- **Enhanced Navigation:** tiles along the route allow more detailed information to be presented along the route without having to increase the payload in the initial call to the routing service. Exits along the highway, names of upcoming roads, and other information can be presented when graph tiles are present.

We're excited with what we've accomplished so far, and look forward to showing you more soon. We plan to launch an open routing service as well as downloadable tiles later in the year. In the meantime though, **have a quick browse of the code (<https://github.com/valhalla>)**, read more about THOR in our **previous blog post (<https://mapzen.com/blog/valhalla-intro>)**, and reach out if something doesn't make sense!

· 22 April 2015 ·



David Nesbitt

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.



Kevin Kreiser

Kevin works on routing at Mapzen but secretly tries to work in all mapping disciplines. Er iss aa Pennsilfaanisch Deitscher.

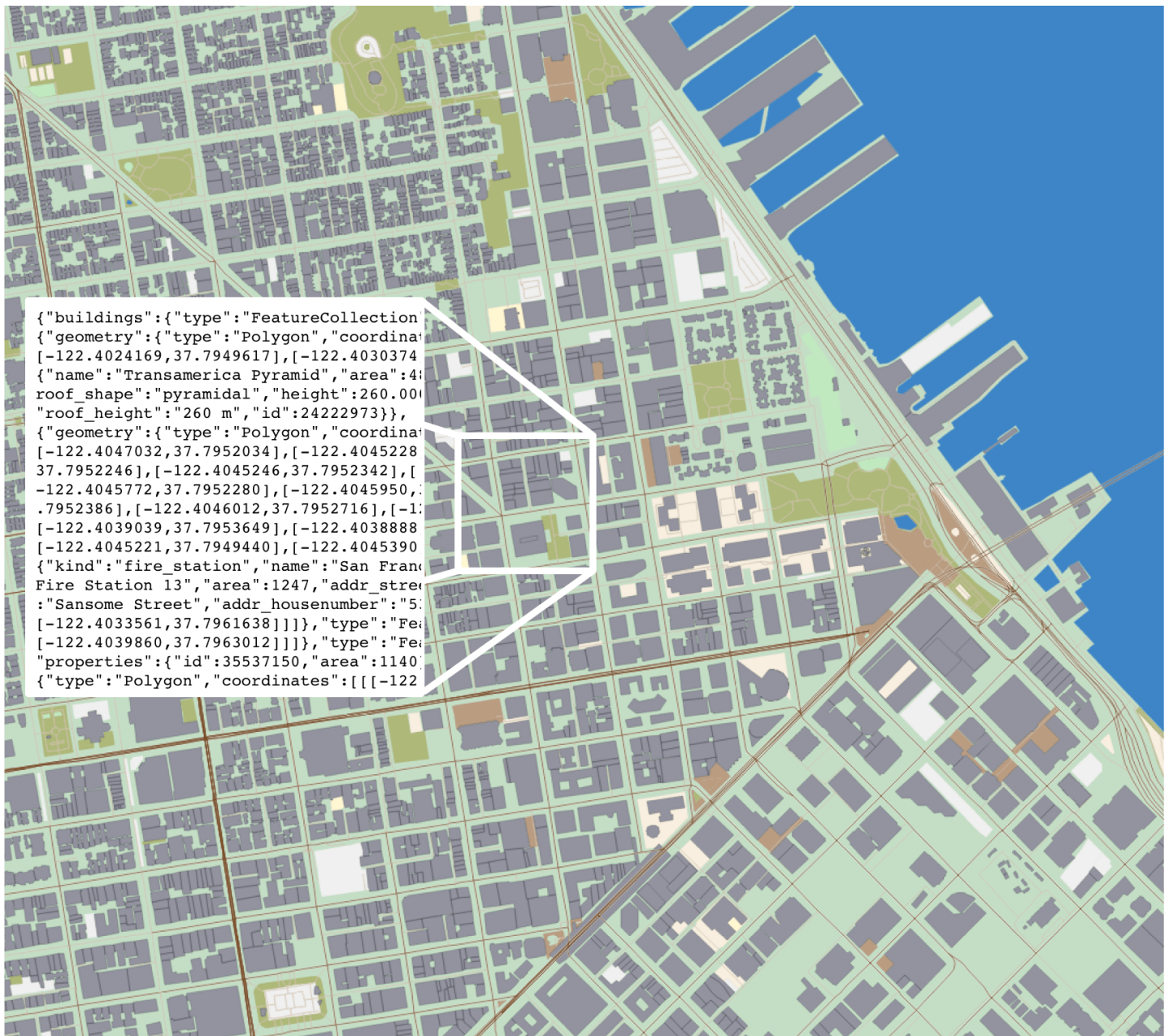
© 2017 Mapzen

Look Upon Our Squares of Math in Three Dimensions

vector-tiles (/tag/vector-tiles) **tangram** (/tag/tangram)

We at Mapzen are proud to announce the launch of our free **Vector Tile Service** (<https://mapzen.com/projects/vector-tiles>) along with **Tangram** (<https://mapzen.com/projects/tangram>), our WebGL map renderer!

Vector tiles are squares of math that describe how to draw roads, buildings, water, and other bits of OpenStreetMap geometry for a particular portion of the earth. A number of browser-based display technologies can consume these vector tiles and display them as the slippy maps you know and love, including D3/SVG, HTML Canvas and WebGL. Vector tiles allow for live updates – no more having to redraw and upload millions of map tiles after making a simple styling change like you had to do in the olden days. Vector tiles are inherently flexible – they can be cached for offline dynamic maps, and can also be human-readable and parsed in any number of ways.



Our Vector Tile service requires an API key, but the good news is it's free! Head on over to **sign up** (<http://mapzen.com/developers>), and take a look at our **Vector Tile page** (<https://mapzen.com/projects/vector-tiles>) to get started.

We are particularly fond of ingesting vector tiles using Tangram. Tangram leverages WebGL and the awesome power of your graphics card to make maps the likes of which **you have only seen in science fiction** (<https://mapzen.com/projects/tangram>)!

Shaders, lighting angles, cameras, materials, styles – Tangram is not your typical map renderer. Head on over to our **Tangram page** (<https://mapzen.com/projects/tangram>) for more details on how to make your own map... *of the future!*

Go **sign up for an API key** (<http://mapzen.com/developers>), check out the **Vector Tile Service documentation** (<https://github.com/mapzen/vector-datasource/blob/master/README.md>), the **Tangram examples page** (<https://github.com/tangrams>), and **let us know what you make** (<https://twitter.com/mapzen>)!

· 12 May 2015 ·



Brett Camper



Rob Marianski

Software engineer working on cutting tiles and infrastructure. Functional programming enthusiast.



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

OpenStreetMap and the Fourth Wall

osm (/tag/osm)

Nokia's **announced intention** (<http://company.nokia.com/en/news/press-releases/2015/04/15/nokia-has-initiated-a-review-of-strategic-options-for-its-here-business>) to sell HERE has sparked a frenzy of speculation around new ownership. Will it be a consortium of **automobile manufacturers** (<http://www.wsj.com/articles/german-car-makers-preparing-formal-bid-for-nokias-here-map-service-with-chinas-baidu-1430843087>)? Will it be **Uber** (http://www.nytimes.com/2015/05/08/business/uber-joins-the-bidding-for-here-nokias-digital-mapping-service.html?_r=0)? Will it be Baidu? Coverage is high on speculation (and tabloid excitement) as the rumored buyers come from many markets and corners of the world.

Two key facts have emerged from the coverage. First, map data is laced throughout many critical industries. From automotive to local search to mobile to business logistics to enterprise business intelligence, map data is a critical asset upon which a huge variety of services are built. Second, there are very few reliable worldwide source of that map data. This combination of wide-ranging importance and scarcity raises the stakes for the eventual owner of the HERE data. And for those who lose access to that data either due to competitive issues or a change in business focus.

What keeps surprising me is how ownership is the main question. I wonder if we're not asking the wrong question. Instead of asking "Who should own this scarce asset?" maybe the right question should be "Why shouldn't this valuable asset be owned by everybody?" If map data is so critical to so many industries, maybe the right answer is that it should be open, supported and used by all.

Google, TomTom, and Nokia HERE keep getting listed as the three spatial data sources available. Why do we forget the fourth one? The one most transparent to the audience, the one we all own and work on together? What about **OpenStreetMap** (<http://www.openstreetmap.org/>)?

OpenStreetMap (<http://www.theguardian.com/technology/2014/jan/14/why-the-world-needs-openstreetmap>) is the invisible fourth wall and it's time to break it. It's time to stop investing billions of dollars in the ownership of an increasingly antiquated dataset and instead invest in leveling the playing field. When digital maps were first created, collecting data about

streets, places, addresses and geography was a specialized task, requiring expensive equipment, sophisticated systems and expert workers. Today, billions of people carry and drive devices that are fully capable of collecting and recording that data every day. Maps are based on physical fact. They are there for anyone to observe, capture, and record. It is time for companies to look at working transparently and in the open so we can turn location into something we all share.

This is the time. The world has never been more ready for an alternative to proprietary map data. People have never been more capable of collecting that data. And governments have never been more interested in pushing their own data out into the public domain. Let's all share map data so we can innovate from a place of experience rather than scarcity and hoarding.

With the Nokia HERE deal supposedly closing at the end of the month, the timing for the largest ever **OpenStreetMap conference** (<http://stateofthemap.us/>) June 6-8th at the United Nations is even more appropriate. We'll be talking about how open geo data can transform an industry and our experiences amidst a radically changing business climate. Whether you represent a company, a government or a local community, we invite you to come and invest in open.

*Alyssa Wright is President of the OpenStreetMap US Board as well as VP of Partnerships at Mapzen. She **originally posted this** (<https://www.openstreetmap.org/user/Alyssa%20Wright/diary>) in her OpenStreetMap diary.*



Alyssa Wright · 18 May 2015

ODIN at SOTMUS

The **State of the Map US conference** (<http://stateofthemap.us/program/>) runs June 6-8 at the United Nations Headquarters in NYC. It's not too late **to sign up to attend** (<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)! The deadline is June 1. Over the next two weeks we will be highlighting the talks by your friends at Mapzen. Duane Gearhart will be discussing Quality Route Guidance on Saturday June 6 at 3:00PM.



Summer's Great Blockbuster has arrived!!

Marvel Avengers?

Mapzen at State of the Map US?

But wait...which one is it?

BOTH

But what do Marvel and Mapzen have in common? Legends like Thor, Odin, Tyr, Loki! Both of these parties have tweaked these Norse Mythology Gods.

Let's examine Odin. Marvel has deliberately softened Odin around the edges. This God of War and Father to Thor and Loki seems to fade into the shadows.

Mapzen has re-invented ODIN - aka *Open Directions Improved Narrative*. At Mapzen, **ODIN** (<https://github.com/valhalla/odin>) is still a wandering God on a quest through the lands, relentlessly seeking and giving wisdom. This one-eyed God is laser focused on improving the quality of directions and guidance.

ODIN will transform path information into real guidance and narrative directions to assist users during their trip. The improved narrative is succinct, easy, and useful. ODIN collapses maneuvers using common base street names and simplifies transitions at complex intersections. Exit and directional information on highways removes ambiguity at key decision points.

So, which blockbuster will you be seeing this summer? If you want to see heroes and Gods save the world (unrealistic) then go to the theater to see the Marvel version. If you want to see Odin save the quality of your trip planning (realistic), then plan on seeing Mapzen at State of the Map US!

*After 15 years as a Mapping Warrior, **Duane Gearhart** (<https://twitter.com/DuaneGearhart>) has been proven to be ever-vigilant, sharp, and analytical on the battlefield, and he has earned his ranks among the Nordic Gods who hail him as the Quality Expert.*

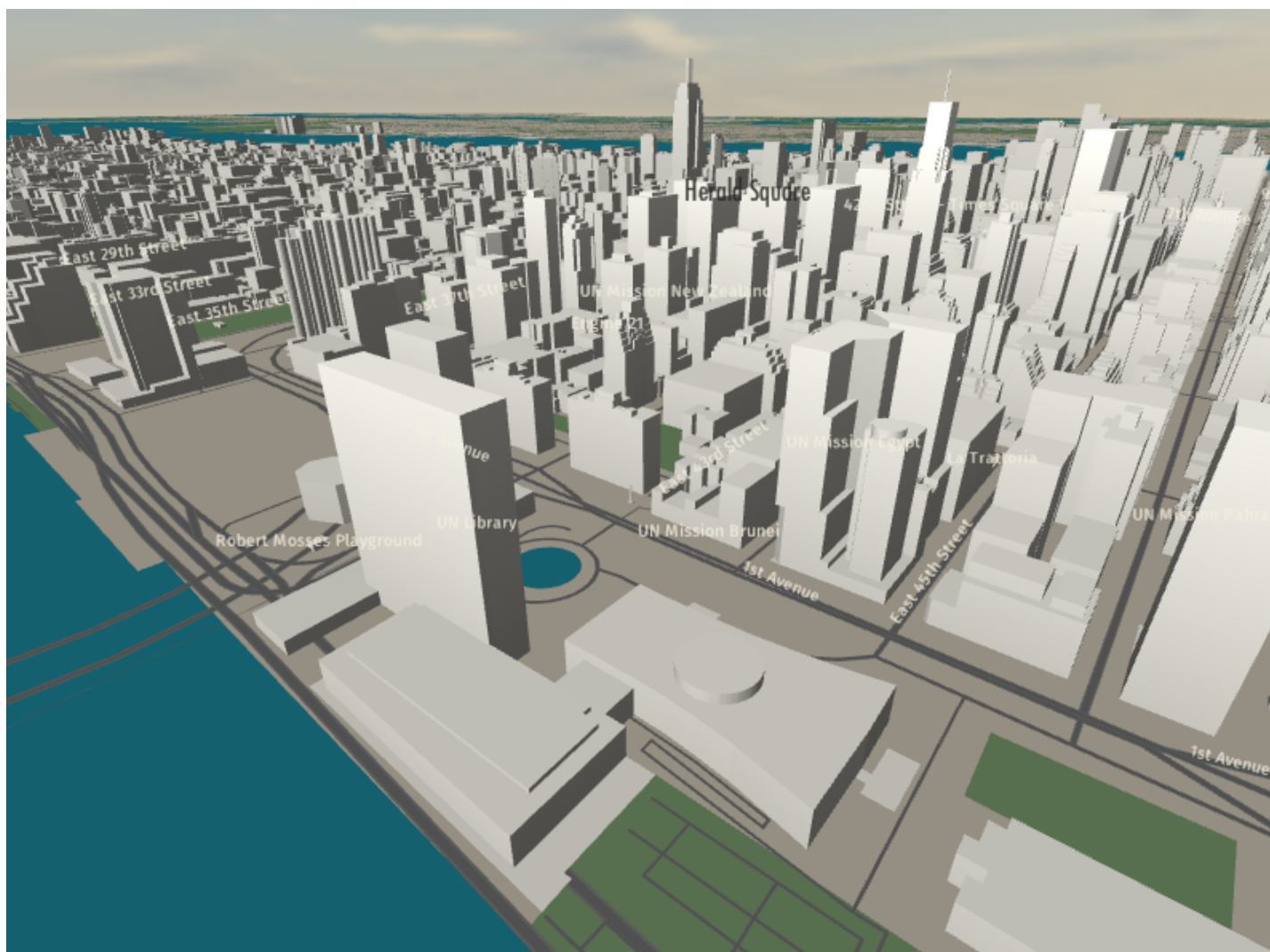


Duane Gearhart · 19 May 2015

Vectorizing Matt Blair

tangram (/tag/tangram) **vector-tiles** (/tag/vector-tiles)

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend** (<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)! Over the next two weeks we will be highlighting the talks by your friends at Mapzen. Matt Blair will be on the **Vector Rendering Panel** (<http://stateofthemap.us/vector-rendering-panel/>) at Saturday 5:00PM in Room CR1.*



Phones and other consumer hardware have become powerful enough to draw beautiful, detailed maps using nothing but vectorized geometry data from OpenStreetMap. We'll discuss techniques and challenges in creating vector renderers that are cross-platform, high-

performance, and really really ridiculously good-looking!

Before joining Mapzen to develop the Tangram ES map renderer, Matt studied robotics at Cornell and made pretentious video games for phones and tablets.

· 21 May 2015

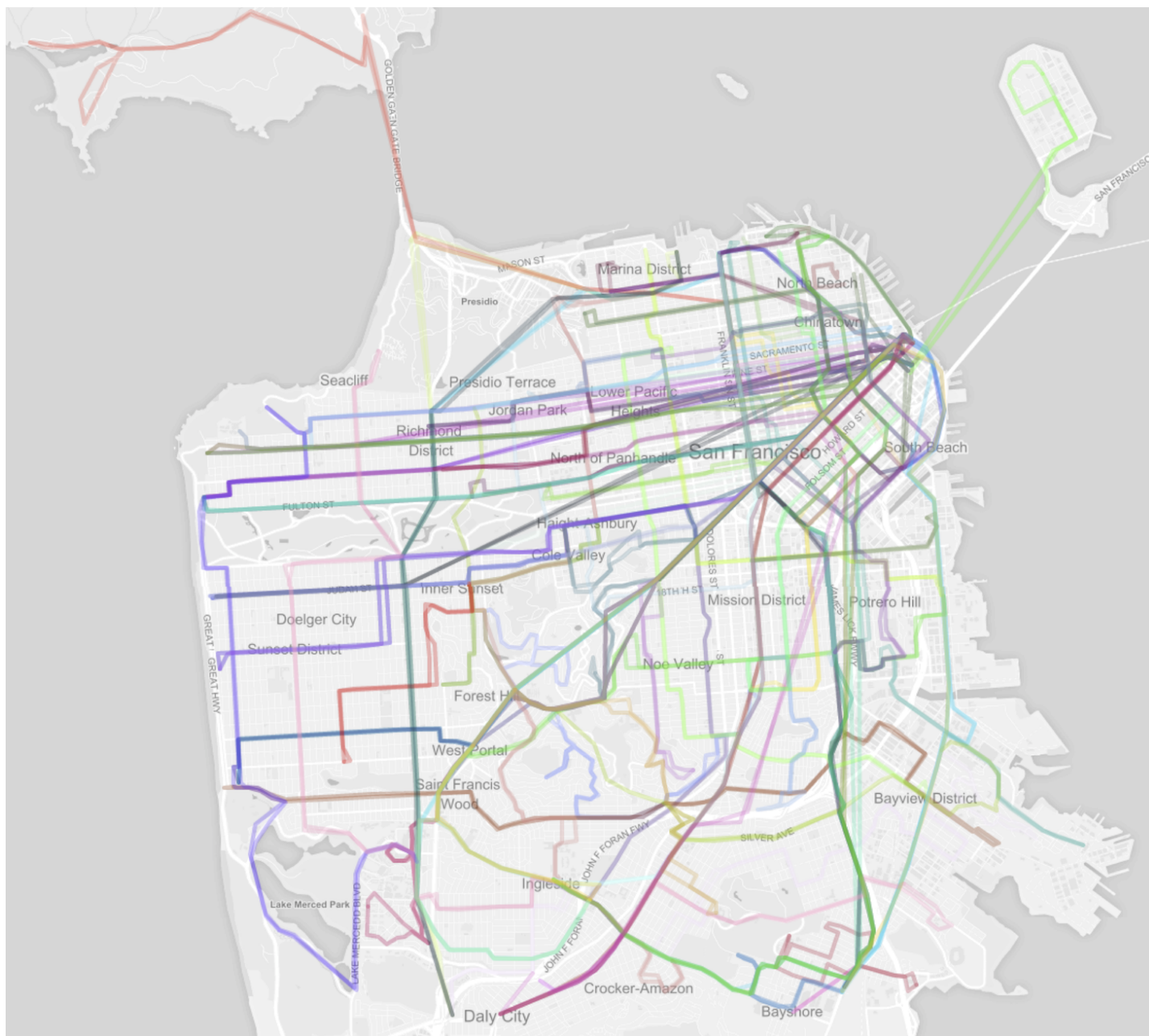
© 2017 Mapzen

Open Transit, Open Sesame

transitland (/tag/transitland)

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend** (<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)! Over the next two weeks we will be highlighting the talks by your friends at Mapzen. Drew Dara-Abrams, Meghan Hade and Ian Rees will conduct an **open transit data workshop on Monday, June 8, 3-4PM** (<http://stateofthemap.us/workshops/>):*

A hands-on tour through open transit data: OpenStreetMap, GTFS, Transitland, and Onestop IDs



OpenStreetMap is full of bus stops, train stations, ferry routes, and even funicular railways, contributed by mappers around the world. Many transit agencies also share their networks and schedules as public data sets, using the General Transit Feed Specification (GTFS) format, created by Portland TriMet and Google. Combining the geographic information of OSM and the temporal information of GTFS feeds opens many of possibilities—but unfortunately it's an often frustrating slog through messy data and incomplete tooling.

In this hands-on workshop, we'll guide participants through a brief tour of working with transit data in OSM and GTFS using open-source tools.

Together, we'll:

- Visually explore transit data
- Use Onestop IDs as a “crosswalk” between GTFS, OSM, the National Transit Database, and other sources of data (to run simple analyses of transit systems)
- Use these as inputs into a multi-modal routing engine (to plan journeys involving transit)

We hope participants will leave with an appreciation of the complexities of OSM and GTFS—and equipped with knowledge and an open-source toolkit to continue their experiments with transit data, software, and ideas.

Drew Dara-Abrams, Meghan Hade, and Ian Rees are on Mapzen’s transit and urban design team, based in San Francisco Bay Area, where there are at least 30 transit agencies.

**Drew Dara-Abrams****Meghan Hade****Ian Rees**

· 27 May 2015 ·

© 2017 Mapzen

Delicious 3D Maps Baked into a RaspberryPi

tutorial (</tag/tutorial>)

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend** (<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)! Over the next few weeks we will be highlighting the talks by your friends at Mapzen. You won't want to miss Patricio Gonzalez Vivo's **Tangram + RaspberryPi Lightning Talk** (<http://stateofthemap.us/lightning-talks-sun/>) on Sunday June 7 at 3:30 pm in Room CR3.*

Open data anywhere for everyone, using OSM data on a RaspberryPi



Make your own GPS device with Tangram ES and RaspberryPi

A couple of months ago the nice folks at RaspberryPi published a **blog post** (<https://www.raspberrypi.org/tangram-an-open-source-map-rendering-library/>) about **Tangram ES** (<https://github.com/tangrams/tangram-es>), Mapzen's native 2D/3D map rendering engine running on their hardware. The feedback was great and people seemed to be very excited to start using it for their own projects.

Tangram ES is a work-in-progress map engine written in C++ and OpenGL ES 2.0, so it can be a little intimidating to start from scratch. That's why we thought this small weekend project could get the ball rolling and ignite some ideas in the community.

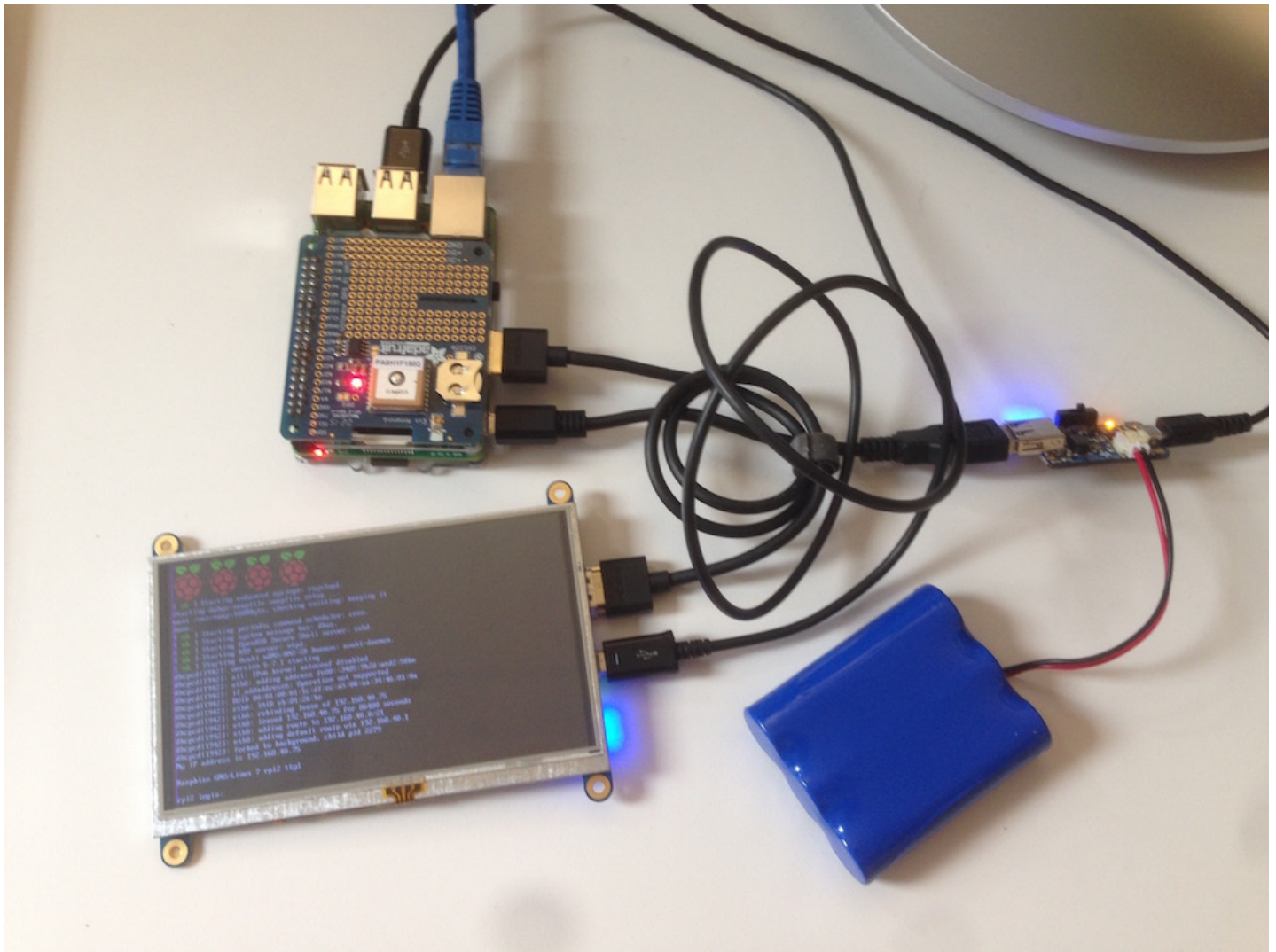
In **this github repository** (<https://github.com/tangrams/PI-GPS>) you will find:

- A 3D-printable model to mount together a RaspberryPi A+/B+, **Adafruit's touch HDMI 5" 800x480** (<https://www.adafruit.com/product/2260>), **Ultimate GPS Raspberry PI HAT** (<https://www.adafruit.com/products/2324>), **Lithium Ion Battery** (<https://www.adafruit.com/products/353>) and **PowerBoost 1000 Charger** (<https://www.adafruit.com/products/2465>).
- Source code to run Tangram ES with a nice little graphical interface to move, rotate and zoom a map with a touch-only screen.
- Instructions for configuring Tangram ES to load tiles locally, using data for your own city or region.

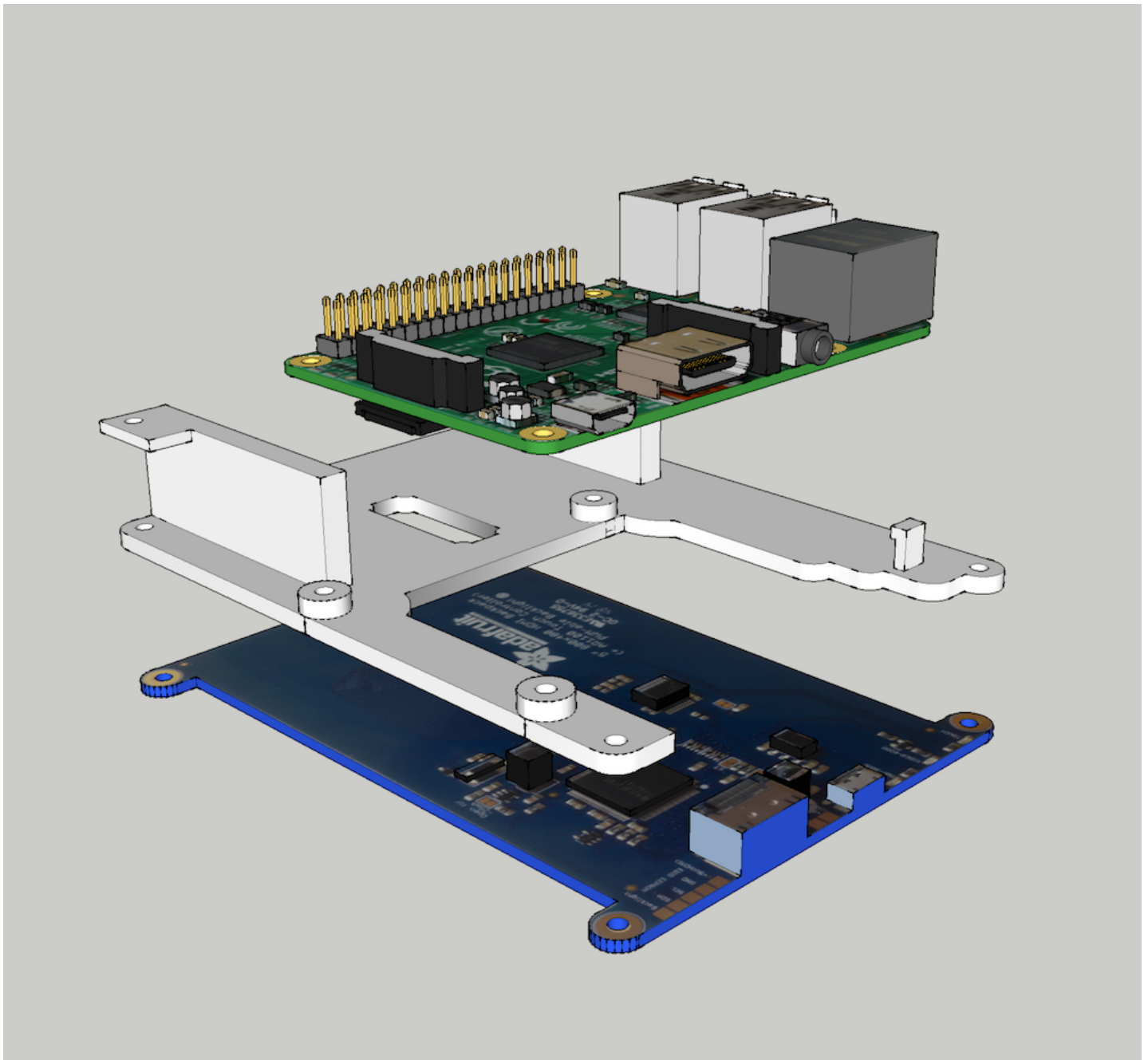
Hardware

Besides your RaspberryPi I'm using the following components to make a stand alone device:

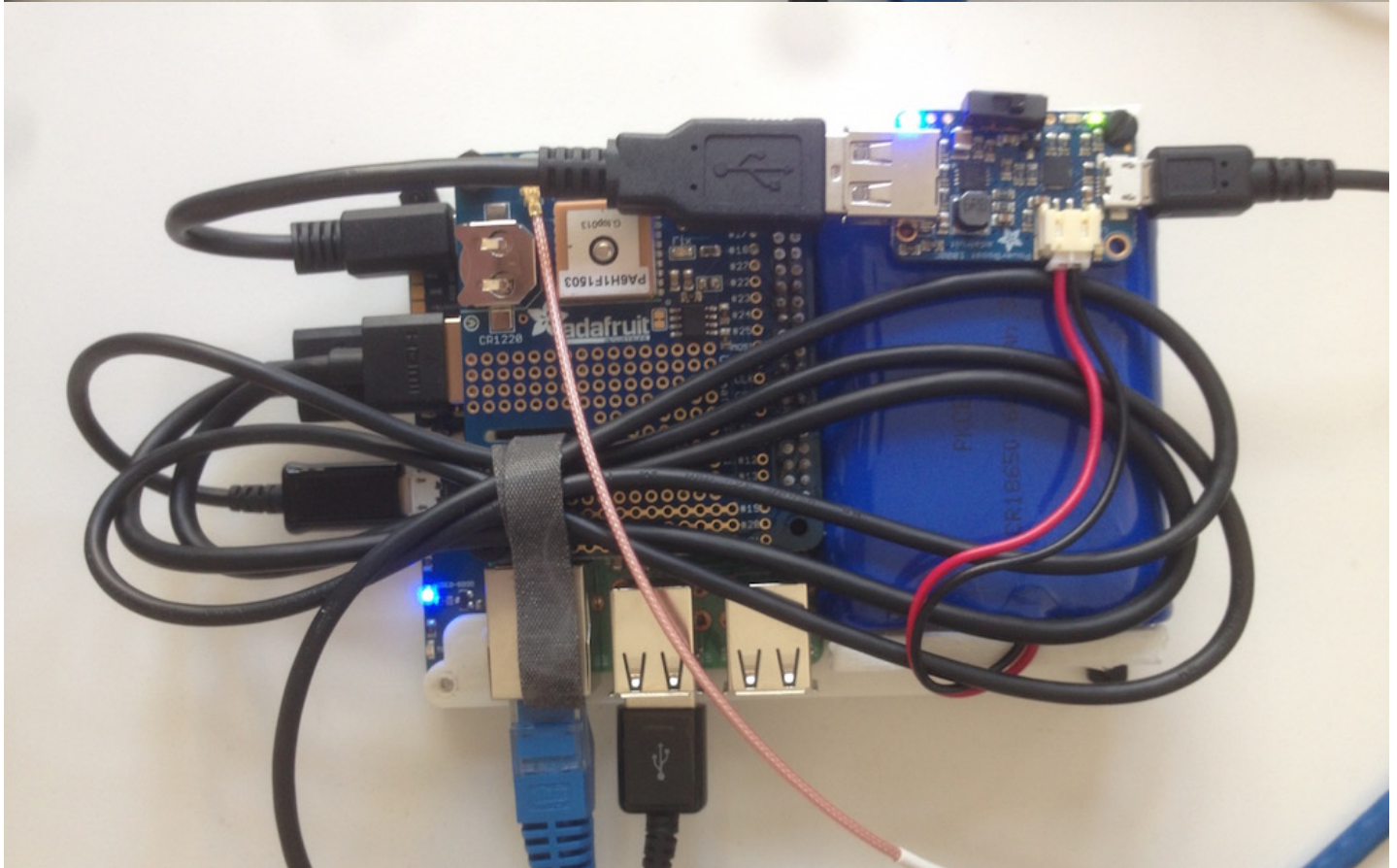
- **HDMI 5 inch 800x480 Touch Screen** (<https://www.adafruit.com/product/2260>)
- **Ultimate GPS Raspberry PI HAT** (<https://www.adafruit.com/products/2324>)
- **Lithium Ion Battery** (<https://www.adafruit.com/products/353>)
- **PowerBoost 1000 Charger** (<https://www.adafruit.com/products/2465>)



These are held together with the 3D-printed mounting station defined in this file: `parts/rpi-screen-mount.stl`



Once you put everything together, it should look something like this:





Compile and install Tangram ES with a nice UI for RaspberryPi

Now let's jump into the code of this project.

First we need to clone this repository in your raspberryPi and install some dependences to compile everything.

```
sudo apt-get update
sudo apt-get install cmake libcurl4-openssl-dev g++-4.8
cd ~
git clone https://github.com/tangrams/PI-GPS.git
cd PI-GPS
git submodule update --init --recursive
mkdir build
```

Then, just to make sure it is working, compile tangram and then run it.

```
export CXX=/usr/bin/g++-4.8
cd build
cmake ..
make
cd bin
./pi-gps -m
```

Note: we are running tangram with the `-m` flag to display the mouse.

Configure Tangram ES to fetch tiles locally

Getting fast internet access to your RaspberryPi could be a problem, especially if you are planning to use this GPS device on your bicycle. Let's see what you need to do to download the map tiles locally making your map network-independent.

We installed Tangram ES and tested it in the previous section. Now it is time for us to make some changes so Tangram ES will search for local files instead of fetching them from a server. Lucky for us, we just need to change that in the configuration YAML file. As with **the web version of Tangram (<https://github.com/tangrams/tangram>)**, this engine is super flexible and you can customize most of it from this file.

Go and open the `~/PI-GPS/tangram-es/core/resources/config.yaml` file, and edit the following line:

```
sources:
  osm:
    type: MVT
    url: http://vector.mapzen.com/osm/all/{z}/{x}/{y}.mvt
```

So it looks like:

```
sources:
  osm:
    type: GeoJSONTiles
    url: file:///home/pi/PI-GPS/build/bin/tiles/{z}-{x}-{y}.json
```

With these changes, tangram will search for tiles inside the `tiles/` directory.

Get the tiles of a town/city/region from OpenStreetMap

My colleague and friend **Peter Richardson (<https://twitter.com/meetar>)** made this useful set of python scripts for downloading tiles locally at the repo called **Landgrab (<https://github.com/tangrams/landgrab>)**.

Let's start by downloading some dependencies for this script.

```
sudo apt-get update
sudo apt-get install wget python-pip
sudo pip install requests
```

Then download Peter's script in the directory where tangram was compiled:


```
cd ~/PI-GPS/build/bin
wget https://raw.githubusercontent.com/tangrams/landgrab/master/landgrab.py
```

Now is the time to download some vector tiles. For that we need the **OpenStreetMap** (<http://www.openstreetmap.org/>) ID of the place we want to download. Go to **OpenStreetMap Website** (<http://www.openstreetmap.org/>) and search for a location and check its ID (the number between brackets).

For example:

- Buenos Aires (1224652)
- London (65606)
- Manhattan (3954665)
- New York City (175905)
- Paris (7444)
- Rio de Janeiro (2697338)
- Rome (41485)
- Sydney (13766899)
- Tokyo (4479121)
- Tucson (253824)

Note: To change the initial coordinates of the map, open `~/PI-GPS/tangram-es/core/resources/config.yaml` again and add a line to the camera block so that it looks like this:

```
cameras:
  camera1:
    position: [-74.00976, 40.705327]
    type: perspective
```

Substitute your own longitude and latitude in the position values. Once we choose a place, tangram will start fetching the vector tiles that we will need. For that we will use the python script we downloaded with the given OSM ID and the zoom level we are interested (in our case all of them are from 1 to 18).

```
python landgrab.py 3954665 1-18
```

This may take a while. Go get a coffee.

Finally! Run it and enjoy your 3D map!

Well done! Everything is ready, unplug your internet connection and run tangram on your RaspberryPi!

```
cd ~/PI-GPS/build/bin  
./pi-gps -m
```

I hope you are excited about all the possibilities of having cool 3D maps on your projects. **Send us (<http://twitter.com/mapzen>)** your opinions, feedback or photos of your work!

Patricio is an engineer and artist making beautiful open source mapping tools at Mapzen. He also teaches in the MFA Design & Technology program at Parsons The New School, and is writing The Book of Shaders, a gentle step-by-step guide to Fragment Shaders. In a previous life he was a psychotherapist and expressive art therapist.



Patricio Gonzalez Vivo · 27 May 2015

© 2017 Mapzen

Extracting Value from SOTMUS

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend—the deadline has been extended to Wednesday, June 3***

(<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)!

*As an incentive, we are highlighting the talks by your friends at Mapzen. You won't want to miss Diana Shkolnikov and Indy Hurt's workshop on **Extracting interesting data from OSM***

(<http://stateofthemap.us/workshops/>) on Monday June 8 at 9 AM.



Ever wonder about the hidden data gems OSM might possess? Feel overwhelmed at the prospect of searching for and extracting them? Well then this workshop is for you. Needle meet haystack no more! We'll take you on a whirlwind tour of a variety of tools available to browse, search, slice, and dice OSM into valuable byte-sized insights.

Diana (<https://twitter.com/dianashk>) is a “classically-trained” software engineer entering the world of all things geo. When she's not busy looking up meanings of geo-related acronyms, she focuses on software design, process and testing all-the-things. She's currently working on an open source geocoder, which she now knows is just a fancy word for geo search.

Indy (<https://twitter.com/indymapper>) is a data scientist lending her geographic expertise to all things “open.” She extracts the good, the bad, and the ugly across time and space to help us understand data in new and exciting ways.



Diana Shkolnikov



Indy Hurt

· 29 May 2015

© 2017 Mapzen

O.S.M.B.A.

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend—the deadline has been extended to Wednesday, June 3***

(<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)!

*As an incentive, we are highlighting the talks by your friends at Mapzen. You won't want to miss Randy Meech speaking about **businesses and OpenStreetMap** (<http://stateofthemap.us/osmba-the-history-and-future-of-companies-in-openstreetmap/>) on Sunday, June 7 at 3PM in CR3.*



Does the OpenStreetMap community have a fraught relationship with businesses? Or have companies just been part of the community since the beginning? Are companies evil institutions that just want to steal all the data, benevolent entities that want to help and contribute, or something in between? We'll review some of the key historical moments of OpenStreetMap and its company ecosystem to figure out how we got to today.

What was Cloudmade and why did so many early OpenStreetMap people work there? What about Skobbler, currently the biggest exit built on OpenStreetMap data? What was the deal with open MapQuest, their services and donations? Why did Steve Coast go to Microsoft, what about their major satellite imagery donation, and what are they doing with OpenStreetMap now? Why would Telenav acquire Skobbler and continue to be involved in the community? How about the rise of Mapbox as the most identifiable OpenStreetMap backed business?

Waze sold for \$1 billion to Google. Is it possible for companies using OpenStreetMap to reach valuations like this? What do companies and their backers think of the license? OpenStreetMap is one of the four global map datasets, what kind of pressure, if any, will this put on the project?

Randy (<https://twitter.com/randyme>) is CEO of Mapzen. Although Mapzen is just under two years old, he's been working on the business side of open mapping for a while. In 2010 he launched **MapQuest's OpenStreetMap initiatives (<http://blogs.wsj.com/digits/2010/07/09/aols-mapquest-looks-to-wikipedia-model-for-mapping/>)**. He's especially interested in how individuals and companies can work together on open data and software.



Randy Meech · 02 June 2015

© 2017 Mapzen

Government and OSM

*The State of the Map US conference at the United Nations Headquarters is in NYC June 6-8. It's not too late **to sign up to attend — the deadline has been extended to Wednesday, June 3 by NOON EASTERN SO HURRY** (<http://www.eventbrite.com/e/state-of-the-map-us-2015-new-york-city-tickets-15437946313>)! As an incentive, we are highlighting the talks by your friends at Mapzen. You definitely won't want to miss Alyssa Wright's panel on **Government and OSM** (<http://stateofthemap.us/osm-and-government-panel/>) on Saturday June 6 at 12:15.*

Is OpenStreetMap just a pretty map for hippy business academics and libertarian developers until the government – our real social structure – takes notice? How is the OSM community weaving this “social fabric”? Are we ready? What type of government rabble rouser is taking notice of the OSM inevitability? And what are the needs, challenges and opportunities that these revolutionaries encounter? Is it just cost? Is it service to the public good? This is a panel that takes OpenStreetMap into our real, civic, and most important world. If you pay taxes, you need to be here.

Alyssa does business at a company that makes things. And she helped plan this conference. Obviously you need to come to the one panel she thought important enough to host.



Alyssa Wright · 02 June 2015

Introducing Valhalla

Mapzen and the Valhalla team are proud to announce the launch of our free routing service! Valhalla is built entirely with open source software that uses Open Street Map data along with other open data sources.



Approaching Valhalla

We have been quite busy getting this service ready and are excited about its potential. We think the basic principles of the Valhalla routing engine are solid:

- *Routing Tiles* that allow memory optimization and future off-line applications
- *Dynamic Costing* that allows flexible, run-time costing to produce routes with different qualities
- *Improved Narrative* that focuses on providing clear and concise guidance along the route path

Feel free to play around with the mini-route demos and peruse the documentation on the **Valhalla project page** (<http://mapzen.com/projects/valhalla/>) to see how this service might work for your application, and **sign up for an API key** (<http://mapzen.com/developers>) to

integrate Valhalla into your maps and apps.

If you are at **State of the Map in NYC this weekend (<http://stateofthemap.us/>)** (June 6-7) you can talk to us about our future plans and maybe even get a peek at what is next. The routing team will be holding a “Routing Power Hour” at the Mapzen booth on Saturday from 3:30-5:30 (ok, well, two hours). If we miss you at State of the Map please check out **our blog posts (<http://mapzen.com/blog>)** and follow us on **Twitter (<http://twitter.com/mapzen>)** to stay up to date with the latest additions and enhancements.

· 04 June 2015 ·



David Nesbitt

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.



Duane Gearhart

Duane is a software engineer @mapzen specializing in quality route guidance and real-world route analysis.



Greg Knisely

Greg is a data munger and software engineer for Mapzen's routing software.



Kristen DiLuca

Kristen is a software engineer specializing in our routing API services. She also enjoys dabbling in javascript for our open source routing test tools and always welcomes new challenges.



Kevin Kreiser

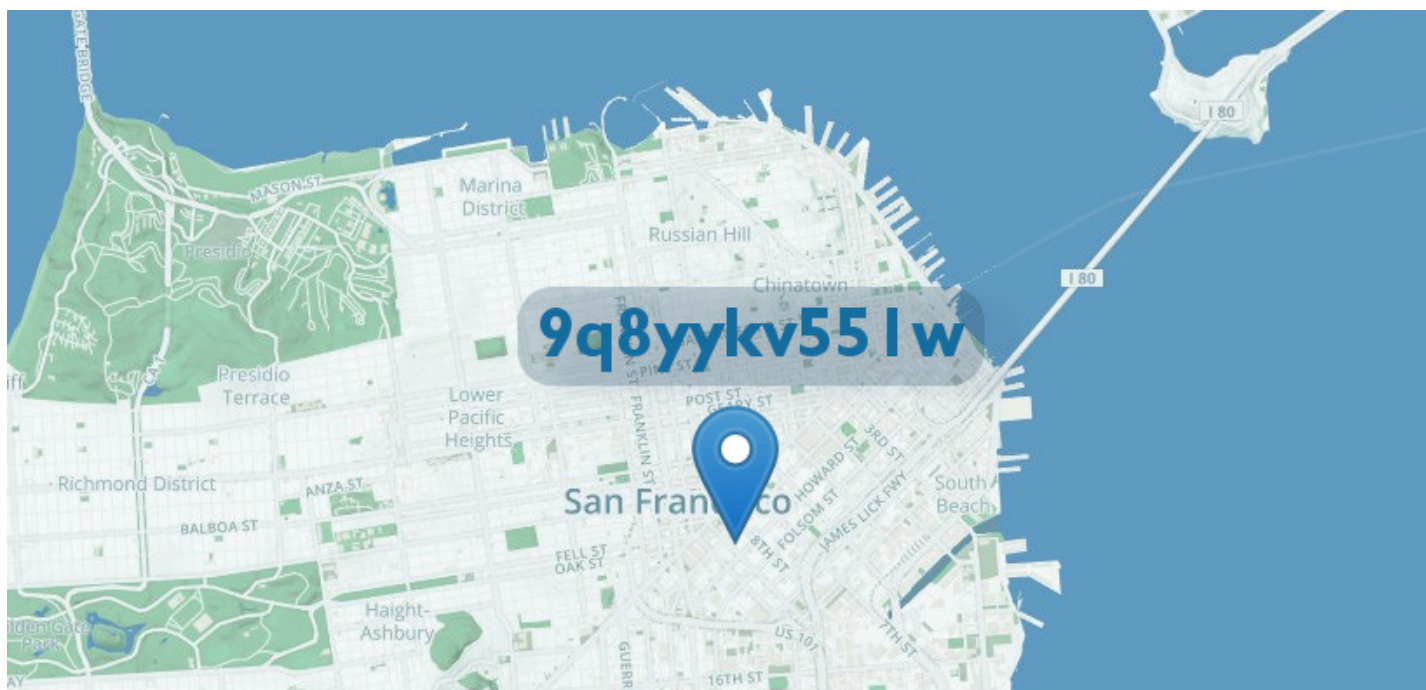
Kevin works on routing at Mapzen but secretly tries to work in all mapping disciplines. Er iss aa Pennsilfaanisch Deitscher.

Geohashes and You

transitland (/tag/transitland)

Today we're introducing Transitland, a community-edited data service aggregating transit networks across metropolitan and rural areas around the world. For a general overview, see **Transitland: Open Transit Data for All** (/blog/transitland-open-transit-data-for-all/).

Geohashes (<http://en.wikipedia.org/wiki/Geohash>), created by Gustavo Niemeyer in 2008 and placed in the public domain, are an elegant and succinct geographic encoding. Geohashes work by reducing a two-dimensional longitude, latitude pair into a single alphanumeric string where each additional character adds precision to the location. Originally created as part of a **URL-shortening service** (<http://geohash.org>), geohashes have proven useful in a variety of contexts, including unique identifiers, spatial indexing, and search.



Transitland (<https://transit.land>) uses geohashes as a prime component of **Onestop IDs** (<https://transit.land/documentation/onestop-id-scheme/>), our system for creating stable identifiers for public transit operators, routes, and stops. The short length and arbitrary precision of geohashes are extremely useful for approximately geolocating millions of transportation locations. For example, the geohash “**9q8znb12j1**”

(<http://geohash.org/9q8znb12j1>)" is the location of the San Francisco Embarcadero subway stop, and forms part of the Onestop ID "s-9q8znb12j1-embarcadero" (<https://transit.land/api/v1/stops/s-9q8znb12j1-embarcadero>)".

Constructing a geohash

The beauty of a geohash is in how it's constructed. In brief, geohashes are a type of grid spatial index, where the world is recursively divided into smaller and smaller grids with each additional bit.



Start with the entire planet, and divide it in half along the Prime Meridian—the first half containing latitudes in the range of $(-180,0)$ as 0 and the second half containing the range $(0,180)$ as 1. The half containing your point becomes the first bit.



Now, divide that half again along the Equator, and add the second bit. Repeat this subdivision, alternating between longitude and latitude, until the remaining area is within the precision desired. Finally, encode the resulting binary sequence using the **geohash base 32 character map** (<http://en.wikipedia.org/wiki/Geohash#Example>) to create the final geohash string. For example, the longitude, latitude coordinate (37.77564, -122.41365) results in the binary sequence "0100110110010001111011110" and produces the geohash "9q8yy".

Binary	01001	10110	01000	11110	11110
Decimal	9	22	8	30	30
Base 32	9	q	8	y	y

Decoding a geohash

Another way of thinking about geohashes are as interleaved longitude (even bits) and latitude (odd bits). This provides a simple method for converting a geohash back into longitude and latitude.

Base 32	9	q	8	y	y
Decimal	9	22	8	30	30
Binary	01001	10110	01000	11110	11110
Longitude	0-0-1	-0-1-	0-0-0	-1-1-	1-1-0
Latitude	-1-0-	1-1-0	-1-0-	0-1-0	-1-1-

As above, each additional bit divides the area in half in a binary search. Latitude begins with a range of (-90, 90) degrees; the first bit of 1 reduces this to (0, 90); the second bit of 0 to (0, 45); the third bit of 1 to (22.5, 45); and so on until with the 12th bit, the range is (37.753, 37.797) with a midpoint of 37.775.

In this case, the width of the area encoded by 12 bits of latitude is +/- 0.022 degrees, or approximately 2.4 km across at the equator. A longer geohash would encode more bits of precision; for example, increasing the geohash from 5 characters to 8 characters would increase latitude to 20 bits, reducing error to +/- 0.00085 degrees, or +/- 0.019 km.

Finding neighbors

As each character encodes additional precision, shared prefixes denote geographic proximity.

City	Geohash	Latitude	Longitude
San Francisco	9q8yym901hw	37.77926	-122.41923
Oakland	9q9p1d5zfs	37.80531	-122.27258
Berkeley	9q9p3tvj8uf	37.86947	-122.27093
Los Angeles	9q5ctr60zyr	34.05366	-118.24276
New York City	dr5regw2z6y	40.71273	-74.00599
London	gcpvn0ntjut	51.50479	-0.07871

City	Geohash	Latitude	Longitude
Greenwich	u10hb5403uy	51.47651	0.00283

You might notice that all the California locations all begin with “9q”, and that Oakland and Berkeley share the “9q9p” prefix. Shared prefixes allow nearby locations to be identified through very simple string comparisons and efficient searching of indexes.

However, it is important to note the converse is not true! The London and Greenwich geohashes are only about 6 km apart as the crow flies—but have no common prefix. This is because the Prime Meridian separates the two locations, dividing these two geohashes from with the very first flipped bit. Fortunately, it is straightforward to calculate the **8 neighbors for a given geohash** (<https://github.com/transitland/mapzen-geohash>). For example, the Southeast neighbor of “gcpv” is “u10h”, successfully jumping across this pesky imaginary boundary.

Searching an area

This method of expanding a geohash to include its neighbors can also be used to efficiently find the set of geohash prefixes to search an area. To find all prefixes within approximately 1 mile of the San Francisco geohash “9q8yykv551w”, first shorten the geohash to 6 characters, “9q8yyk”, then add in all 8 neighbors: 9q8yy7, 9q8yyt, 9q8yy5, 9q8yys, 9q8yym, 9q8yyj, 9q8yyk, 9q8yyh, 9q8yye. Any point in this area is known to start with one of these prefixes.



At Transitland, we use this property to specify simple bounding boxes for transit operators as part of a Onestop ID. For example, the **SFMTA** (<http://www.sfmta.com/>) has over **3500 bus stops** (<https://transit.land/api/v1/stops?servedBy=o-9q8y-sfmta>). As described above, it is quite possible that there is no common geohash prefix shared by every stop in the system (or, if there is, it can be so short as to be of limited usefulness). In the case of the SFMTA, only “9q” prefix is shared in common, which covers a very large portion of the Southwestern United States. However, the geohash prefix “9q8y”, when expanded to include neighbors, provides the 9 prefixes that can be used to find any stop, while covering a much smaller land area. This “neighbors implied” geohash is then used in the Onestop ID for SFMTA: “f-9q8y-sfmta” (https://transit.land/api/v1/operators?onestop_id=o-9q8y-sfmta).

Start where you are

Try the **Transitland Playground** (</blog/welcome-to-the-transitland-playground>), a simple and attractive interface to browse transit operators, stops, and routes, including their Onestop IDs and geohashes.

Browse a **worldwide map of geohashes overlaid on a “slippy” webmap** (<http://mapzen.github.io/leaflet-spatial-prefix-tree/>).

Use our **Python library** (<https://github.com/transitland/mapzen-geohash>) to compute geohashes and their neighbors.

And do keep in touch

To read even more about how we're looking at the world's transportation data, find us at **transit.land** (<https://transit.land>) and **@transitland** (<https://twitter.com/transitland>)

Updates

Transitland API and documentation links updated on May 3, 2016.

· 05 June 2015 ·



Ian Rees

Ian spends too much time thinking about cities, land use, and open transit data on the Mapzen Mobility team.

© 2017 Mapzen

Transitland: Open Transit Data for All

transitland (/tag/transitland)



A COMMUNITY-EDITED DATA SERVICE



AGGREGATING



TRANSIT NETWORKS

ACROSS METROPOLITAN



AND RURAL

AREAS AROUND THE WORLD.



(<https://transit.land>)

Hello from the San Francisco Bay Area, where half of the ground is heading northbound and the other is inching southward. (Remember from elementary school? That's what causes earthquakes.) Also epically disjoint is our mass transit: **over 30 public agencies** (<http://www.spur.org/publications/spur-report/2015-03-31/seamless-transit>), an ever-increasing number of **private shuttles** (<http://stamen.com/zero1/>), not to mention **jitneys** (<http://www.yelp.com/biz/jess-losa-jitney-97-san-francisco>) and **carpools** (<http://sfcasualcarpool.com/>) for those in the know.

At Mapzen, we're trying to make sense of the jumble here in the SF Bay Area—and in the many other parts of the world with puzzle-like transit systems—using open-source software and open data. The name of our new effort: **Transitland** (<https://transit.land>).

Transitland aggregates existing geographic and temporal data from authoritative sources, connecting together common records using common identifiers. And with Transitland, we're equipping ourselves, our collaborators, and perhaps also you, to edit and improve this "community datastore" with knowledge from boots on the ground. It's with these station locations, agency identifiers, and route schedules that we can join together the "tectonic plates" of mass transit in the SF Bay Area and around the world.

Every piece of Transitland is open. We welcome you to use the data and the tools to build your own apps, gadgets, visualizations, journey planners, and analyses.

Start where you are

Are you a transit enthusiast or planner, interested in browsing the range of data available in Transitland? **Read about our Playground (/blog/welcome-to-the-transitland-playground)** and give it **a try (https://transit.land/playground)**.

Are you a software developer, interested in Transitland's tooling? **See our "how it works" diagram (https://transit.land/how-it-works/)**, with links to GitHub repos.

Are you a transit data expert, interested in how Transitland joins feeds, operators, stops, and routes from disparate sources? Read more about **our Onestop ID scheme and its most important ingredient, the geohash (/blog/geohashes-and-you)**.

And do keep in touch

Join us at **transit.land (https://transit.land)** and **@transitland (https://twitter.com/transitland)**

· 05 June 2015 ·



Drew Dara-Abrams

Drew leads Mapzen Mobility products (and aspires to being a flâneur).

© 2017 Mapzen

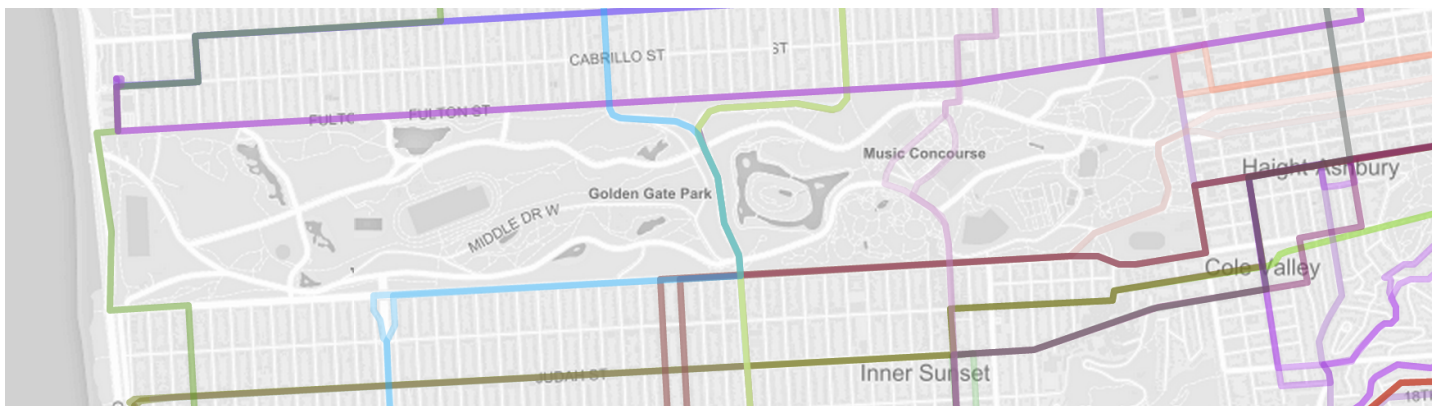
Welcome to the Transitland Playground

transitland (/tag/transitland)

*Today we're introducing Transitland, a community-edited data service aggregating transit networks across metropolitan and rural areas around the world. For a general overview, see **Transitland: Open Transit Data for All** (/blog/transitland-open-transit-data-for-all/).*

Transitland's (<http://transit.land>) mission is to make the world's transportation data as accessible as possible. We do this by smoothing out some of the quirks of the **General Transit Feed Specification (GTFS)** (<https://developers.google.com/transit/gtfs>) format commonly used by transit agencies and then aggregating the transit data into a centralized, accessible place called the **Transitland Datastore** (<https://transit.land/how-it-works/#/1>).

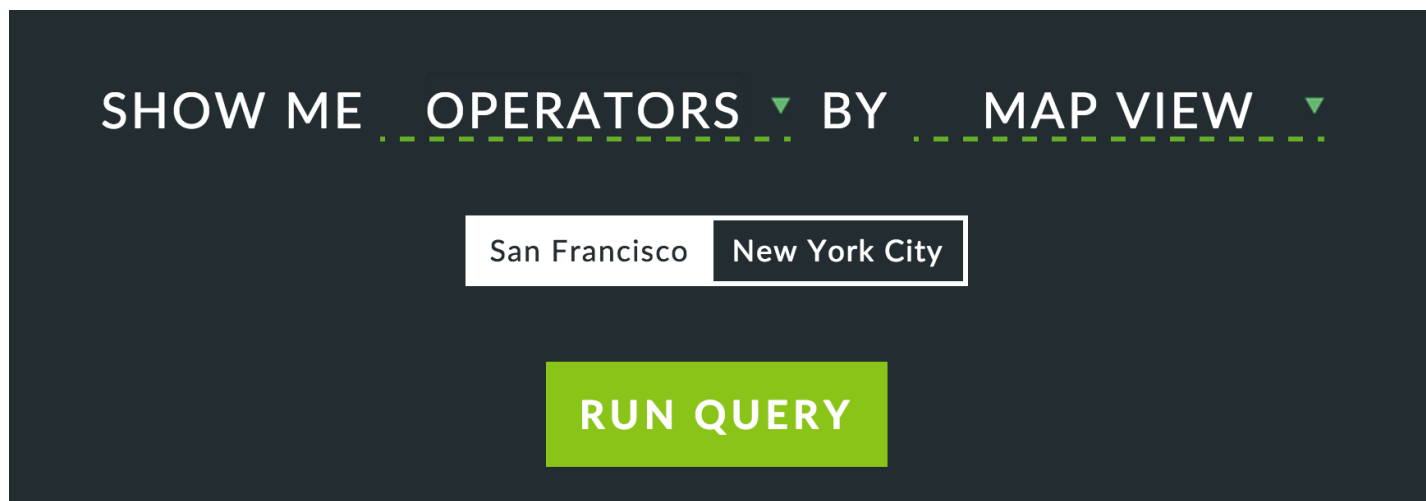
For those of us who would like to work with transit data but are not programmers—urban planners, GIS analysts, advocacy groups, or **Maptimers** (<http://maptime.io/>), for example—knowing where to start with a web service like the Datastore might seem daunting. This is where the **Transitland Playground** (<https://transit.land/playground/>) comes in. The Playground acts as a front door to the **Datastore API** (<https://github.com/transitland/transitland-datastore#api-endpoints>), providing a way to view and download all of the data without writing any lines of code.



(<https://transit.land/playground>)

The Playground offers a simple way to visualize transit data. Curious about transportation options near a park in your city? No need to visit numerous transit provider websites, download multiple datasets, or mentally conflate various transit system maps just to understand how it all

fits together. Using the Playground, you can zoom the map to the area of interest, run a query, and see what's there.



(<https://transit.land/playground>)

The Playground uses a “Mad Libs” style to search the Datastore for operator, route, and stop information. Currently, there is data available for six transportation providers in the San Francisco Bay Area and two in New York City, with more on the way. The results are downloadable as well, in CSV and JSON formats (the JSON data download is a bookmarkable page, so you can save your search results). This allows you to work with the data however you’d like—create dynamic web maps of transit systems, run GIS analyses on the neighborhoods around transit hubs, advocate for transit improvements in your city... we’re excited to see how you explore the data!

Start where you are

Ready? **Start exploring the Transitland Playground** (<https://transit.land/playground>).

And do keep in touch

Join us at **transit.land** (<https://transit.land>) and **@transitland** (<https://twitter.com/transitland>)

· 05 June 2015 ·

Meghan Hade



Meghan is a software engineer with a background in urban planning on Mapzen's Mobility team. She works in javascript and plays in calligraphy, block printing, and finding ways to stash n+1 bikes in a San Francisco apartment.

© 2017 Mapzen

Avoiding the Total Perspective Vortex

osm (/tag/osm)

Being new to OpenStreetMap, I couldn't help but be reminded of the Total Perspective Vortex in the classic Hitchhiker's Guide.

The Total Perspective Vortex derives its picture of the whole Universe on the principle of extrapolated matter analyses. To explain – since every piece of matter in the Universe is in some way affected by every other piece of matter in the Universe, it is in theory possible to extrapolate the whole of creation – every sun, every planet, their orbits, their composition and their economic and social history from, say, one small piece of fairy cake.

The man who invented the Total Perspective Vortex did so basically in order to annoy his wife.

Trin Tragula – for that was his name – was a dreamer, a thinker, a speculative philosopher or, as his wife would have it, an idiot. And she would nag him incessantly about the utterly inordinate amount of time he spent staring out into space, or mulling over the mechanics of safety pins, or doing spectrographic analyses of pieces of fairy cake.

“Have some sense of proportion!” she would say, sometimes as often as thirty-eight times in a single day.

And so he built the Total Perspective Vortex – just to show her.

And into one end he plugged the whole of reality as extrapolated from a piece of fairy cake, and into the other end he plugged his wife: so that when he turned it on she saw in one instant the whole infinity of creation and herself in relation to it. To Trin Tragula’s horror, the shock completely annihilated her brain; but to his satisfaction he realized that he had proved conclusively that if life is going to exist in a Universe of this size, then the one thing it cannot afford to have is a sense of proportion.

Douglas Adams, The Restaurant at the End of the Universe, Chapter 11

Downloading the planet-latest file felt like peeking into the vortex. As OSM continues to gain momentum, it gets harder and harder to maintain a sense of proportion without adverse effects on one’s sanity. So if a dataset of this size is to exist, we simply cannot afford to have a sense of proportion. We must allow for a variety of helpful slivers of the planet to help us represent its entirety in a variety of situations.

As already established, OSM is massively, unapologetically HUGE(ly awesome)!!! We all want, love, take part in, and encourage this. Consequently, this leads to us having less and less insight into what is actually in there. It's hard to gauge coverage, consistency, and accuracy of the data when said data represents so many things to such a diverse community. It is on that diverse community then, to extract valuable subsets of the planet and focus on the quality of that subset independent from the rest of the data.

Mapzen has done just that, and we have chosen administrative boundaries as a starting point. This decision was inspired by the prior work of David Blackman with Foursquare – you can learn more about his efforts here:

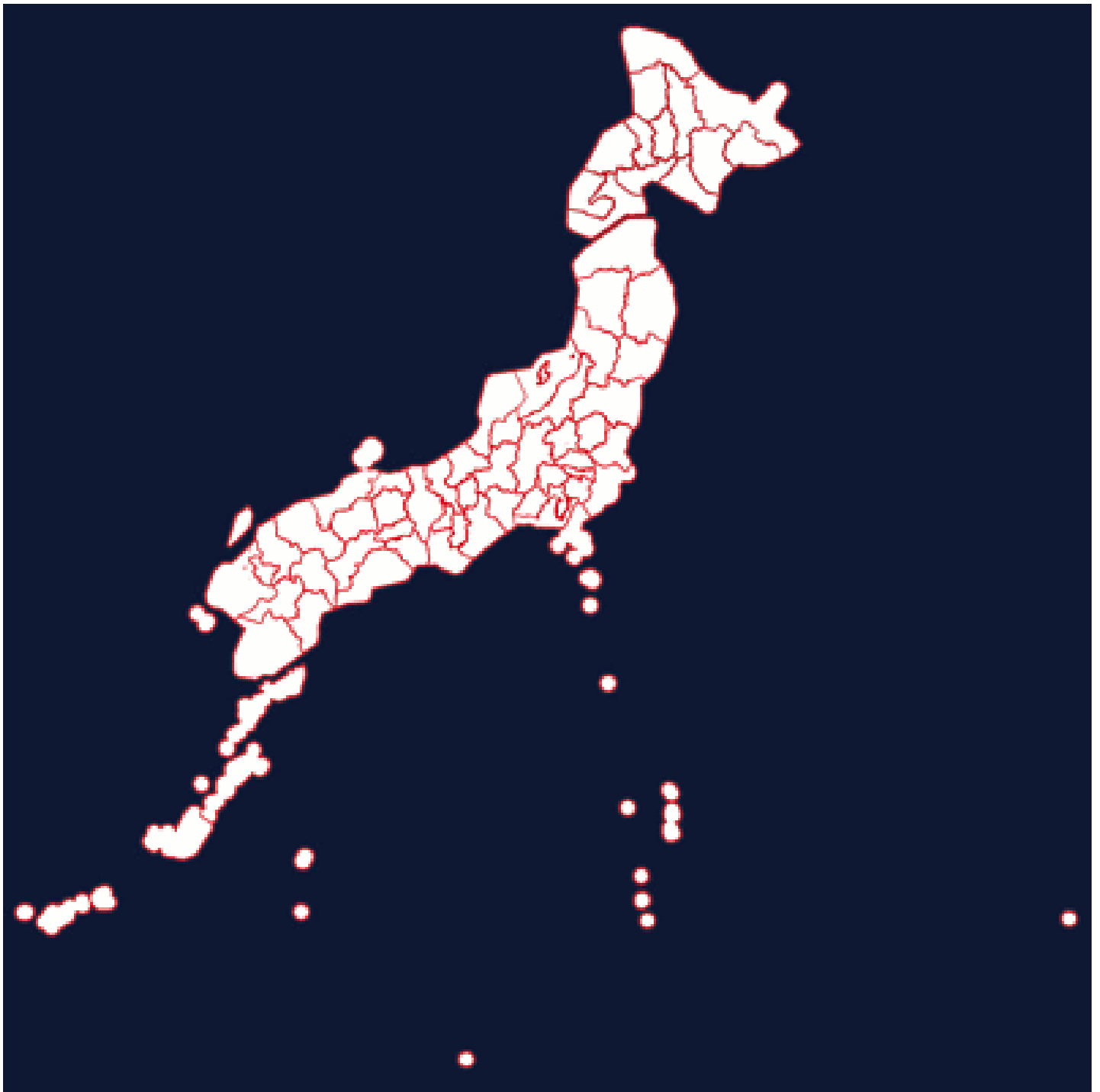
- <https://vimeo.com/91893151> (<https://vimeo.com/91893151>)
- <http://quattroshapes.com/osm> (<http://quattroshapes.com/osm>)

Administrative boundary subset of OSM data has tremendous value in a variety of geo problems, so it seemed like a great place to start.

So what have we done exactly? We used `osmfilter` to extract all relations that have a combination of `"boundary"="administrative"` and `"admin_level"=*` tags. You can read all about the meaning of these tags on the **OSM wiki** (<http://wiki.openstreetmap.org/wiki/Tag:boundary%3Dadministrative>). We then threw those relations into the clutches of `osmium` and converted each one to MultiPolygons in GEOJSON files. The planet extract contains a file per `admin_level` value, for example `admin_level_2.geojson` as well as `admin_level_other.geojson` for all the polygons where `admin_level` value was non-numeric.

To take it a step further, we decided to also slice that planet boundary data up into manageable country chunks, since often geo problems are highly localized and only require data for a specific country. We had to somehow programmatically decide which countries should be extracted, so why not use the planet boundaries we just extracted? All it took was grabbing the `admin_level_2.geojson` from the planet borders, which indicates countries, and pulling out anything that had a `flag` tag. This isn't perfect, but it works surprisingly well, and is what some might call "eating our own dogfood".

This GIF shows the increasingly detailed admin boundaries of Japan, from `admin_level 2` to `7`. (There are three more, running down to the town and neighborhood level, but they are not readily visible at this scale.)



We've made this data publicly available for download at **mapzen.com/data/borders** (**<http://mapzen.com/data/borders>**) in the hopes that it will help people in need of such data. More importantly however, we hope that exposing this data will drive the community back to OSM to improve its quality. In order to highlight some of the problems we encounter during the extract process, we've included an errors.json file in the planet extract. So if a border you're looking for isn't present, check the errors to see if there is something you can do in OSM to fix it. We will continue to run the extraction process monthly to ensure data improvements are reflected.

In addition to the extracts being available for download, we've also made the tools used to generate these extracts open-source and available here:

- <https://www.npmjs.com/package/fences> (<https://www.npmjs.com/package/fences>)
- <https://github.com/pelias/fences-cli> (<https://github.com/pelias/fences-cli>)

Stay tuned for future posts on how to use the tools to make custom region extracts. As always, we welcome contributions from the community.

This border extract example is just the beginning -- it's the beginning of a conversation about valuable subsets of goodness hiding within OSM data. These data sets need attention in order to make them complete, accurate, and all around awesome. Sets such as neighborhoods, postal codes, and coastlines are just a few of the other potential extracts to consider. We'd love to hear what you think is important to extract out of OSM and improve as a community.

I should mention that there is one concern to us gaining insight. According to the Hitchhiker's Guide, *"There is a theory which states that if ever anyone discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable."* Let's hope that's not the case with OpenStreetMap!

· 06 June 2015 ·



Diana Shkolnikov

Diana is the engineering director of Search@Mapzen and a proponent of mandatory fun, testing, education, and paleo, not necessarily in that order.

© 2017 Mapzen

State of the Map US 2015 Wrapup



We at Mapzen would like to thank the organizers of State of the Map US for a such great conference. The United Nations was an amazing venue, and it was a pleasure to see so many old friends and make so many new ones in the talks, at the social events and at our booth. We are

fortunate to be part of a community that is so enthusiastic about building things with OpenStreetMap.



Here's a summary of the Mapzen talks as well as links to our video and transcript archives:

Saturday (June 6) Talks

Government & OpenStreetMap: Landscapes, Perspectives and the Horizon
(<http://stateofthemap.us/osm-and-government-panel/>), moderated by Alyssa Wright

video (<https://www.youtube.com/watch?v=2hZLa0Ln-Yk>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#alyssa)

"I hear a lot of comments about adopting OpenStreetMap in government agencies. We've found that it really is much more cost-effective to improve the data. Hiring the students and spending that first year improving it was less expensive for us than licensing proprietary data for just a one-year period." —Bibiana McHugh, Trimet

Peripheral Data in OpenStreetMap (<http://stateofthemap.us/peripheral-data-in-osm-panel/>), **Drew Dara Abrams, Diana Shkolnikov**

video (<https://www.youtube.com/watch?v=5BN600ukexw>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#drew-diana) | **presentation** (<https://github.com/mapzen/presentations/blob/master/06-2015-SOTMUS/PeripheralDataInOpenStreetMap.pdf>)

"Isn't OSM big enough for everything under the sun? So shouldn't there be room for everything within OSM proper? OpenStreetMap data model is very flexible, but there are other types of objects it doesn't represent very well." —Diana Shkolnikov

Quality Route Guidance (<http://stateofthemap.us/quality-route-guidance/>), **Duane Gearhart**

video (<https://www.youtube.com/watch?v=hwglqOV6l9M>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#duane) | **project page** (<https://mapzen.com/projects/valhalla>)

"So this is Odin, the one-eyed god. He's focused on improving route guidance." — Duane Gearhart

Vector Rendering Panel (<http://stateofthemap.us/vector-rendering-panel/>), **Matt Blair**

video (<https://www.youtube.com/watch?v=0RA5k7J5Sw0>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#matt)

"I would say [vector map rendering is] freedom... a way of democratizing style." — Matt Blair

Mapzen is building Fences...and knocking down barriers (<http://stateofthemap.us/lightning-talks-sat/>), **Diana Shkolnikov**

video (https://www.youtube.com/watch?v=Q_6DlaxMU0Q) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#diana) | **project page** (<https://mapzen.com/data/borders>)

"So we all know that OSM is massively unapologetically hugely awesome...But as it continues to grow, we consequently have less and less idea of what's actually in there. We know that at the end, when you collect all this data, it's going to be amazing, and everyone is going to win, right? But we have to figure out what to do in the interim, until it becomes awesome." —Diana Shkolnikov

Sunday (June 7) Talks

State of the Geocoder (<http://stateofthemap.us/state-of-the-geocoder/>), **Harish Krishna** | **project page** (<https://github.com/pelias>)

video (<https://www.youtube.com/watch?v=mLb8-QveXGo>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#harish)

"I'm pretty sure that everybody in this room has built a Geocoder at some point. I know that you're in denial. You've thought about it." —Harish Krishna

Improving Diversity in OpenStreetMap (<http://stateofthemap.us/improving-diversity-in-osm/>), **Kathleen Danielson**

video (<https://www.youtube.com/watch?v=WlzTEaMEc8k>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#kathleen) | **Ada Initiative** (<https://adainitiative.org/>)

“We spend a lot of time tolerating bad behavior, and it’s by being very passive – we are making our priorities super, super clear. So let’s change those priorities and make that super clear. Stop relying on invisible labor. Stop relying on people who already don’t have access to the same resources to build up an understanding of why they don’t have access to these resources.” —Kathleen Danielson

O.S.M.B.A. The history and future of companies in OpenStreetMap (<http://stateofthemap.us/osmba-the-history-and-future-of-companies-in-openstreetmap/>), **Randy Meech**

syllabus (<https://github.com/randymeech/osmba-syllabus>) | **video** (<https://www.youtube.com/watch?v=MgEh9d9Rj74>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#randy)

“I would like for all of us to work together. A community of individuals and businesses, to make [OpenStreetMap] the best data set in the world, using our shared resources, which then moves competition away from the data. Let’s just share this stuff. There’s so much more interesting stuff to do.” —Randy Meech

Open data anywhere for everyone, using OSM data on a RaspberryPi (<http://stateofthemap.us/lightning-talks-sun/>), **Patricio Gonzalez Vivo**

video (<https://www.youtube.com/watch?v=Jg2m7jM-gg8>) | **transcript** (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#patricio) | **project page** (<https://github.com/tangrams/PI-GPS>)

“I think true openness means lowering the technology bar, to make this more accessible for everybody.” —Patricio Gonzalez Vivo

Monday (June 8) Workshops

Extracting interesting data from OSM (<http://stateofthemap.us/workshops/>), Diana Shkolnikov and Indy Hurt

“So here’s the question. I found something interesting. How do I actually get this out of OpenStreetMap and create a map with it?” —Indy Hurt

pdf (<https://github.com/mapzen/presentations/blob/master/06-2015-SOTMUS/ExtractingInterestingThingsWorkshop-Indy-Diana.pdf>) | transcript (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#indy-diana)

A hands-on tour through open transit data: OpenStreetMap, GTFS, Transitland, and Onestop IDs (<http://stateofthemap.us/workshops/>), Drew Dara-Abrams, Meghan Hade, Ian Rees

transcript (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#drew-meghan-ian) | project page (<https://transit.land/>)

“I think everyone should have access to the data to use in their own way, without necessarily being a developer. And those types of people – advocacy groups, planners, community groups – that’s who the [Transitland] playground is for.” — Meghan Hade

Geocoder-in-the-box (<http://stateofthemap.us/workshops/>), Peter Johnson

transcript (https://mapzen-assets.s3.amazonaws.com/images/state-of-the-map-2015-wrapup/sotmus2015_Mapzen_transcripts.html#peter) | project page (<https://github.com/pelias>)

“If you want to build your own geocoder, what sort of features would you be looking for? You want it to be Open Source, because it’s so complex. There’s so many different edge cases and different countries in the world, different address schemas, all that sort of stuff, that you can’t be there on the ground to observe the truth and to ensure the accuracy of your product. So it has to be Open Source, unless you have a hundred million... Billions of dollars that you can go and spend and put people on the ground to ensure its quality.”

· 18 June 2015 ·



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

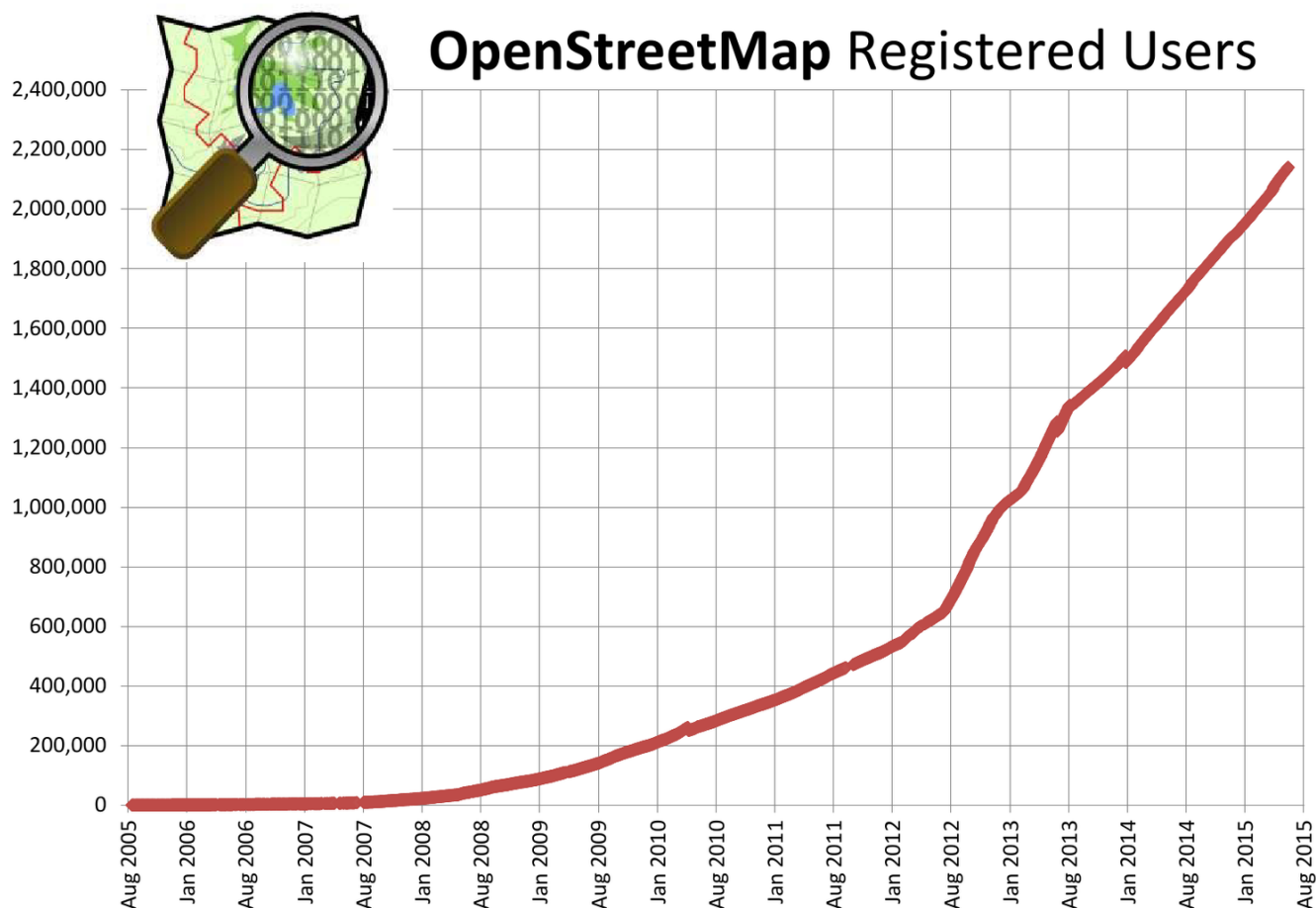
OSM Server-thon

osm (/tag/osm)



*UPDATE: the server funding target has been achieved! Thanks to all the individual contributors – over 1200! And don't forget that OpenStreetMap **always welcomes donations** (<http://donate.openstreetmap.org/>).*

OpenStreetMap is popular! Almost too popular! The OSM Foundation has announced its 2015 hardware fundraising drive, and new servers are on list. The existing OSM servers are projected to hit capacity all too soon, so **please donate today!** (<http://donate.openstreetmap.org/server2015/>)



There are **more OSM registered users than ever** (<http://wiki.openstreetmap.org/wiki/Stats>) and every dollar and pound and euro and yen counts. Mapzen and our friends at Mapbox have donated, and we hope you can too. The **details of the upgrade requirements** (<http://donate.openstreetmap.org/server2015/#the-upgrade>) give a sense of just what amazing work the Operations Working Group has been doing, **and how we can help them continue to do so.** (<http://donate.openstreetmap.org/server2015/>)



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

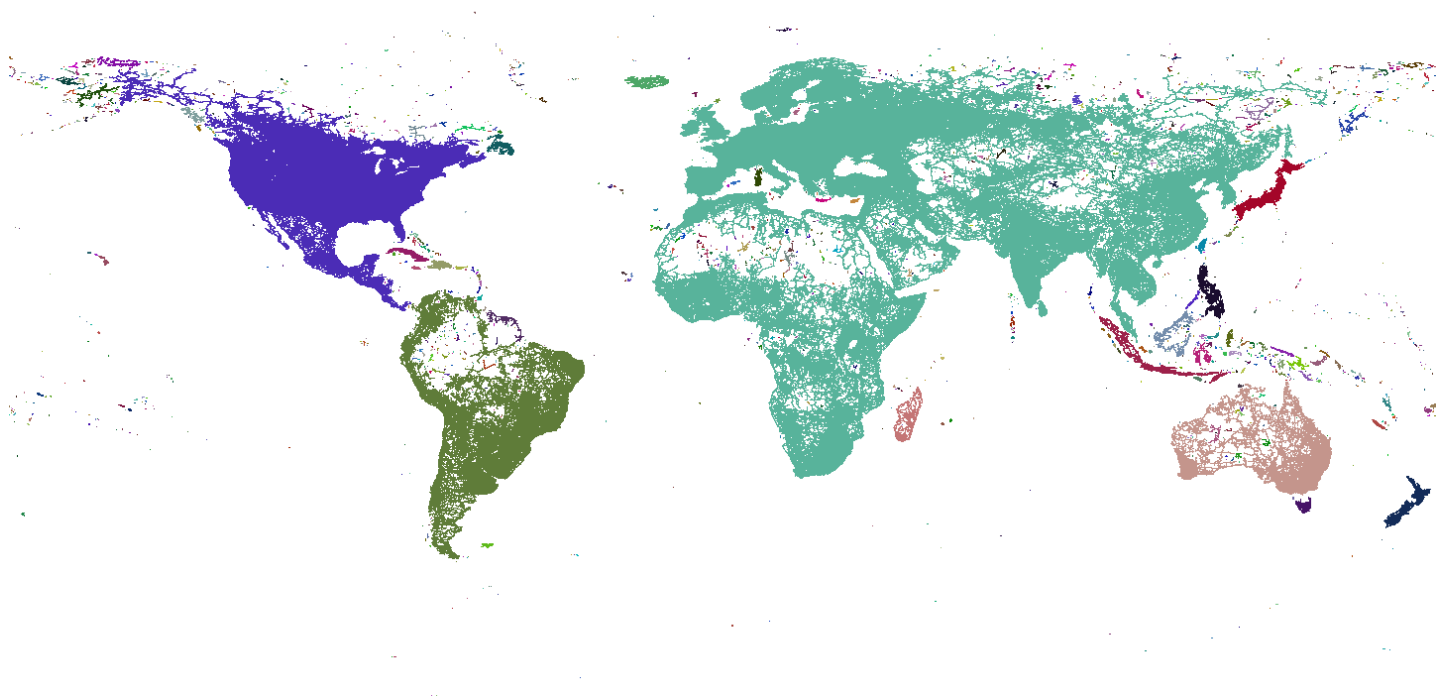
You, Me and Connectivity

routing (/tag/routing)

OSM Routing Connectivity Map

In our work on open source routing, the Valhalla team at Mapzen have found an interesting byproduct of producing routing tiles - they provide a rough, first order approximation of connectivity.

Valhalla graph tiles are a uniformly sampled grid in latitude, longitude space. A Valhalla graph tile exists wherever there are navigable ways. Conversely no graph tile exists if there are no navigable ways within a given tile's region. Imagine each tile in the grid as a pixel in an image. It just so happens we have indeed used these tiles to create such an image, in which regions (tiles) of the same color are connected by OSM ways. (Note that any uncolored region (white in the image) is deemed unreachable via Valhalla's various modes of transportation.)



Steps to Produce the Connectivity Map

Using only the files that contain the Valhalla graph data, we can create the connectivity image using a simple flood fill algorithm:

1. Iterate through tiles on disk to figure out which exist, placing each into a hash map as the key with an initial color of white (uncolored)
2. Iterate over the entries in the map
 - If this entry is uncolored
 - Select a new color and create a queue of only the tile in this entry
 - While the queue is not empty
 - Pop a tile off the queue
 - Color it and add any of its neighbors to the queue who appear in the map
3. Iterate over all the tiles in the world to create an image in PPM (Portable Pixel Map) format (use white for any tiles that are not in the map)
4. Convert the PPM image into png using ImageMagik
5. Flip the image vertically (Valhalla tiles are row ordered from South to North).

GeoJSON

For all you geo hackers out there, we also provide a **connectivity map in geojson format** (<https://s3.amazonaws.com/mapzen.valhalla/prod/connectivity.geojson>). The colors in the geojson feature properties are simply integers so you'll want to do something on your own to get a proper color out of that for rendering.

Dealing with Ferries and Long Ways

Valhalla stores routing data in tiles where there are nodes that intersect more than one OSM way or are endpoints of an OSM way. Some OSM ways may be long and have no intersecting ways along the path (e.g. ferries). Normally this would result in no Valhalla tiles along that path. Without special processing, these long ways would create a connectivity map that falsely shows no connectivity between the endpoints of the way. To get around this, we create empty graph tiles (no nodes or edges) to indicate there is indeed connectivity through that tile's region.

Using the Connectivity Map

The connectivity map has one very important use within Valhalla - we are instantaneously able to reject routes that have no possible connection. If any 2 locations lie in tiles of different "colors" (or if either are white) in the connectivity map, then they have no possible path between them over navigable ways. A quick image lookup allows us to efficiently filter these impossible paths. (Note that just because 2 locations have the same color does not mean they are guaranteed to have a valid, connecting path.)

We hope that this map will also be useful to the OSM community to identify regions where no connectivity exists but perhaps it should, allowing for focused efforts to map regions and improve OpenStreetMap road data.

· 23 June 2015 ·

**Kevin Kreiser**

Kevin works on routing at Mapzen but secretly tries to work in all mapping disciplines. Er iss aa Pennsilfaanisch Deitscher.

**David Nesbitt**

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.

© 2017 Mapzen

Our Magna Carto

One of my favorite pages on Wikipedia lists the **oldest companies in the world** (https://en.wikipedia.org/wiki/List_of_oldest_companies). While mapping is indeed an ancient profession, you won't find any mapping companies there: the oldest companies make alcoholic beverages, knives and swords, or provide restaurant and hotel services. The oldest I've done business with is probably Stella Artois (1366).

The oldest company in the world was until recently **Kongō Gumi** (https://en.wikipedia.org/wiki/Kong%C5%8D_Gumi), a Japanese construction company founded in 578. Yet even good old Kongō Gumi ran into difficulties in 2006 and was absorbed into a larger construction company, proving that nothing in this world is permanent. Even **Shitennō-ji Buddhist temple** (<https://en.wikipedia.org/wiki/Shitenn%C5%8D-ji>), which it originally built in the 6th century, has been completely rebuilt a number of times over the centuries:



(<https://en.wikipedia.org/wiki/Shitenn%C5%8D-ji#/media/File:Shitennoji07s3200.jpg>)

Shitennō-ji: younger than it looks. image credit 663highland, CC BY 2.5

I started in tech in 2000, and moved into the mapping industry in 2008. Not a long time ago by any means, although feeling *like I've seen things before* happens more frequently to me at this phase of my career. And what I've seen over the years is change and impermanence. Giants have tripped, hot startups have been acquired and fallen quite literally off the map. What has remained and been carried forward?

Seeking permanence

One of the core tenets of Mapzen is that all our software and data must be open. This has been our guiding principle since the beginning, and as sure as all things must end, we will follow this through to our end (hopefully a hundred or more years from now, like Kongō Gumi). By building everything in the open we can build for real permanence, not just the short-lived permanence of a hot brand in this turbulent space.

Our motto is “Start where you are.” This means many different things, but I often think about it when we’re looking at an area where open data is clearly inferior to proprietary, such as geocoding. A conventional company would abandon its open philosophy and strike a deal with a proprietary geocoding service. But **to quote Google**

(<https://investor.google.com/corporate/2004/ipo-founders-letter.html>), we are not a conventional company: we will start where we are and insist on open data. This is part of our work with Code for America and OpenAddresses: **we would rather spend \$500,000 improving open address data (<http://www.codeforamerica.org/blog/2015/03/03/mapzen-and-our-work-on-geographic-open-data/>)** than strike a quick deal to provide a better geocoder. We would rather do things that benefit all in the open.

Why? Because striking a quick deal robs from our collective future. It props up old, struggling business models, and prevents us and our developers from using and improving data as a community. It’s not that we don’t have the money to work with traditional map data providers. We won’t work with them because it would divert focus from the real work of improving open data, which will help everyone. We will never take harmful shortcuts, even if they make short-term business sense.

At the US State of the Map conference, I spoke on **businesses and OpenStreetMap** (<https://www.youtube.com/watch?v=MgEh9d9Rj74>). At the end of the talk I invited the whole community - individuals, businesses, governments - to work together to get all of this data into the open. Doing this will allow us to build something that will outlast many things, and will certainly outlast the older proprietary models that we see collapsing around us now. Then we can build much more interesting things on top.

As companies and individuals, let's all work on things that will be around for the next hundred years.

· 01 July 2015 ·



Randy Meech

Billerica Memorial High School graduate. BA, MTS. Mapzen CEO 2013-present.

© 2017 Mapzen

Use Metro Extracts in QGIS

osm (/tag/osm)



Map tiles by **Stamen Design** (<http://stamen.com>), under **CC BY 3.0**
(<http://creativecommons.org/licenses/by/3.0>) | © OSM contributors

Are you still doing your own processing just to extract an area of interest from the global OpenStreetMap dataset? You can put your computing efforts toward other tasks because **Metro Extracts** (<https://mapzen.com/data/metro-extracts/>) provides OpenStreetMap data clipped to the size of your city or region.

You're intrigued by Metro Extracts, but could use help getting started. Which Metro Extracts formats are available and which one should you download? How can you effectively filter the data into a usable map? Get answers to these and many other questions in this new tutorial, *Extract OpenStreetMap data for display in QGIS*.

Mapzen built **this tutorial** (<https://mapzen.com/documentation/metro-extracts/walkthrough/>) to walk you through the process of choosing from the available formats and loading the extracted files into **QGIS** (<https://www.qgis.org/en/site/forusers/download.html>), a free and open-source desktop GIS application. In the exercises, you will use QGIS to explore the contents of Metro Extracts files, including viewing the OpenStreetMap attributes, performing queries on the data, and changing the symbols used to represent the features. It should take you about an hour to work through the tasks and build a map.

As part of Mapzen's **commitment to open source and open data** (<https://mapzen.com/blog/our-magna-carto>), the documentation for Mapzen's projects are in public GitHub repositories, along with the software code. The Metro Extracts tutorial is in the **mapzen/metroextractor-cities** ([https://github.com/mapzen/metroextractor-cities](https://github.com/mapzen/metroextractor-cities/tree/master/docs)) repository. If you have any comments on the steps, you can drop a note as a GitHub issue, or fork it and add your own exercises.

You can find the tutorial at <https://mapzen.com/documentation/metro-extracts/walkthrough/> (<https://mapzen.com/documentation/metro-extracts/walkthrough/>).

This post was updated on April 27, 2016.

· 21 July 2015 ·



Rhonda Glennon

Rhonda is Mapzen's technical publications manager and writes about maps and developer tools.

OSRM Sunset



After a strong ride of nearly a year, we are sunsetting our OSRM routing service over the next few months. We encourage our OSRM users to **sign up for a free API key** (<https://mapzen.com/developers>) and **try Valhalla** (<https://mapzen.com/projects/valhalla>).

The **Open Source Routing Machine** (<http://project-osrm.org/>) is a great example of what can be accomplished with open tools and open data, and we were glad we could provide it as a free public service for those who didn't want to set up their own servers **or cross the streams** (<https://mapzen.com/routing#loc=15,40.7467,-73.9937>).

After working with OSRM, we soon realized that routing with open source data could be made even better by embedding road characteristics and connectivity information in tiles, along with providing dynamic, run-time costing for custom transportation modes. In under six months, the Valhalla development team came on board and created a **open source** (<https://github.com/valhalla>), world-class routing and navigation tool using OpenStreetMap road network data.

Some of the benefits of Valhalla include narrative directions, collapsed maneuvers, simplified transitions at intersections, and better exit information. And that's not all: in the future we plan on tile downloading and multimodal routing.

We already have more users and routes generated on Valhalla than our OSRM service – **Remix** (<http://getremix.com/>) (formerly TransitMix) switched with no difficulty soon after Valhalla API was released in June.

Just upgraded our routing engine to the new hotness from **@mapzen** (<https://twitter.com/mapzen>) – Valhalla. Definitely the new standard in routing <https://t.co/562klH9qjK> (<https://t.co/562klH9qjK>)

— Remix (@remixcities) **June 15, 2015**
(<https://twitter.com/remixcities/status/610480534825754624>)

How easy is it to switch? Queries to the API are formatted in JSON rather than as arguments, and the output format is slightly different, but these are all explained in the **Valhalla API documentation** (<https://github.com/valhalla/valhalla-docs/blob/master/api-reference.md>).

GaiaGPS was able to quickly build an **GPS trail finder** (<http://blog.gaiagps.com/plot-a-precise-trail-with-the-gaia-gps-trail-finder/>) using Valhalla, which is proving popular in the hiking community:

@gaiagps (<https://twitter.com/gaiagps>) So much better than any existing route planner for hiking, and was quick to build using open open services & open data.

— Jesse Crocker (@DataMongers) **July 10, 2015**
(<https://twitter.com/DataMongers/status/619549531160444928>)

Our friends **@gaiagps** (<https://twitter.com/gaiagps>) have raised the bar (and then some) on trail route creation. Awesome! <https://t.co/uBCLIt0NQU> (<https://t.co/uBCLIt0NQU>)

— Brad Winney (@bradwinney) **July 13, 2015**
(<https://twitter.com/bradwinney/status/620644369679282176>)

And if you are using Leaflet, we now have a **plugin** (<https://github.com/valhalla/lrm-valhalla>) for the **Leaflet Routing Machine** (<http://www.liedman.net/leaflet-routing-machine/>)! You can install it using npm:

```
npm install lrm-valhalla
```

Further instructions are available on **Github** (<https://github.com/valhalla/lrm-valhalla>) and will be on the Leaflet Routing Machine **alternative routers page** (<http://www.liedman.net/leaflet-routing-machine/tutorials/alternative-routers/>) soon.

If you have any questions, please contact us at routing@mapzen.com (<mailto:routing@mapzen.com>).

· 27 July 2015 ·



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

Make your first Tangram map

tangram (/tag/tangram) **tutorial** (/tag/tutorial)



Beautiful cartography. Animation. Shading. 3D buildings showered in **glitter** (<https://tangrams.github.io/tangram-sandbox/tangram.html?styles/specular-dust.yaml#16/40.7053/-74.0098>)? Yes! All these and many other design techniques are possible with Mapzen's Tangram map rendering engine.

Be on your way to authoring 2D and 3D maps with Tangram by following a new **tutorial** (<https://github.com/tangrams/tangram-docs/blob/gh-pages/pages/walkthrough.md>). In the lesson, you will build a simple Tangram map and learn how to change its appearance by modifying a few lines of text.

A human-readable text file (a scene file), organizes all the styling elements needed to draw a Tangram map. Editing the scene enables fine control over almost every aspect of cartography, including symbols, colors, lighting, and feature labels.

After you update the scene file and finish styling your Tangram map, you will learn how to upload it to the web using the free **GitHub Pages** (<https://pages.github.com/>) hosting service, allowing you to share your work with others.

You can find the tutorial at <http://git.io/vYBiM> (<http://git.io/vYBiM>).

· 29 July 2015 ·



Rhonda Glennon

Rhonda is Mapzen's technical publications manager and writes about maps and developer tools.

© 2017 Mapzen

Intern Report—Ladders for Leaders

interns (/tag/interns)

Hello!

I'm Elizabeth, and I've been interning at Mapzen's NYC office for the past six weeks. I discovered Mapzen and its internship program through the Summer Youth Employment Program's **Ladders for Leaders** (<http://www.nyc.gov/html/dycd/html/jobs/ladders.shtml>) which helps provide professional internships to NYC youth.

Before starting here, I had no experience with maps or open source, and had only just begun programming a few months ago. I had no idea about the kind of tasks I'd be working on, and had no working experience with any of the tools and languages mentioned on the internship listing. Though learning about new coding languages, tools, and frameworks for each task was daunting, it gave me the opportunity to learn a lot over a short period of time.

Most of what I've done is related to Mapzen's website. I worked on pages that displayed statistics on developers and projects, where I gained working experience in Ruby on Rails, HTML, CSS, and Javascript. I also added a new feature to blog posts so that you can read a brief author biography beneath each post. Look out for that soon!

During my last week, I focused on a new project. Using Leaflet, I created a map using a Tangram layer that displays data from the Gazetteer, an open, comprehensive data set of the world that describes countries, regions, cities, neighborhoods, and their relations to one another.



This map started as update of **Aaron Cope's classic but defunct "I Am Here" map** (<http://www.aaronland.info/iamhere/#latitude=37.7756922&longitude=-122.41410579205677&zoom=14&style=2241>) which was one of the first back in olden times to make it easy to copy the latitude and longitude of the center of a sloppy map. In addition to dynamically updating the lat/lon, our map shows the names of the neighborhoods, localities, and regions as the user drags the map. (It can also jump to the user's current location, or a city from a predefined list.) Although still in its early stages, this framework will eventually allow us to check for and add more location names, and find out where we're missing data.

Beyond programming, I've also learned more about the mapping community, open source, and open data, and its importance. I've had plenty of fun working with the Mapzen team too! My summer at Mapzen was definitely both invaluable and enjoyable.

· 17 August 2015 ·



Elizabeth Williams

Elizabeth studies computer science at Stony Brook and was a NYC SYEP Ladders to Leaders intern at Mapzen. She built this blog blurb system and thinks burritos are pretty good.

© 2017 Mapzen

Who's On First

whosonfirst (/tag/whosonfirst)

The short version



All the places! Still not a magic pony!! Yet!!!

The longer version

Mapzen is building a gazetteer of places. Not quite *all* the places in the world but a whole lot of them and, we hope, the kinds of places that we mostly share in common. A gazetteer is a big list of places, each with a stable identifier and some number of descriptive properties about that location. An interesting way to think about a gazetteer is to consider it as the space where debate about a place is *managed* but not decided. We call our gazetteer “Who’s On First” (or sometimes “WOF” for short).

According to **Wikipedia** (https://en.wikipedia.org/wiki/Who's_on_First%3F), Who's on First:

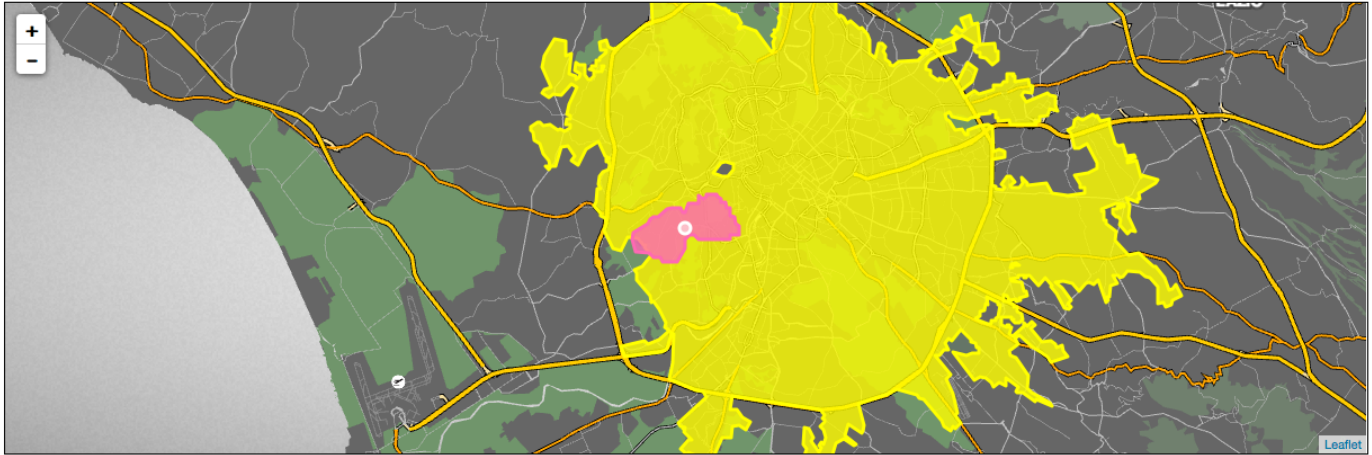
...is a comedy routine made famous by Abbott and Costello. The premise of the sketch is that Abbott is identifying the players on a baseball team for Costello, but their names and nicknames can be interpreted as non-responsive answers to Costello's questions. For example, the first baseman is named "Who"; thus, the utterance "Who's on first" is ambiguous between the question ("Which person is the first baseman?") and the answer ("The name of the first baseman is 'Who'"). "Who's on First?" is descended from turn-of-the-century burlesque sketches that used plays on words and names. Examples are "The Baker Scene" (the shop is located on Watt Street) and "Who Dyed" (the owner is named Who). In the 1930 movie *Cracked Nuts*, comedians Bert Wheeler and Robert Woolsey examine a map of a mythical kingdom with dialogue like this: "What is next to Which." "What is the name of the town next to Which?" "Yes." In English music halls (Britain's equivalent of vaudeville theatres), comedian Will Hay performed a routine in the early 1930s (and possibly earlier) as a schoolmaster interviewing a schoolboy named Howe who came from Ware but now lives in Wye.

Which sort of sums up the "problem" of geo, nicely. It might be easier, perhaps, if we all understood and **experienced the world as coordinate data** (<https://github.com/google/open-location-code>) but we don't, so the burden of "place" and its many meanings is one we trundle along with to this day.

Our gazetteer is absolutely not finished – both in terms of data coverage as well as data quality – so, in the near-term, you should adjust your expectations accordingly when you approach the data.

We are releasing the data now because *we believe it is important not just to articulate our goals and intentions around the project but also to back them up with tangible proofs*. Think of this blog post, and the data we are publishing today, as the arc and direction of our efforts but not necessarily where they will land.

Villa Doria Pamphili



Properties

```
{u'geom:area': u'0.001089',
 u'geom:bbox': u'12.396311,41.861379,12.462004,41.892055',
 u'geom:latitude': u'41.876621',
 u'geom:longitude': u'12.428392',
 u'gn:id': u'',
 u'iso:country': u'IT',
 u'misc:gn_adm0_cc': u'IT',
 u'misc:gn_id': u'0',
 u'misc:gn_local': u'3169070',
 u'misc:gn_nam_loc': u'Roma, Roma, Roma, Lazio, IT',
 u'misc:gn_namadm1': u'07',
 u'misc:local_max': u'52663',
 u'misc:local_sum': u'239886',
 u'misc:localhoods': u'28',
 u'misc:name': u'Villa Doria Pamphili',
 u'misc:name_adm0': u'Italia',
 u'misc:name_adm1': u'Lazio',
```

Villa Doria Pamphili is a **neighbourhood** and its consensus geometry is from [quattroshapes](#)

Hierarchy

Ancestors that Villa Doria Pamphili is parented by

- the country of [Italy](#)
- the region of [ROMA](#)
- the locality of [Roma](#)

Other

- [See all the descendants of Villa Doria Pamphili](#)
- [Reverse geocode](#) (41.876621, 12.428392)

```
{}
```

We believe it is important not just to articulate our goals and intentions around the project but also to back them up with tangible proofs.

Ours is not the first gazetteer (and hopefully not the last). Many gazetteers have existed before. Notable examples include:

- The Getty Institute's **Thesaurus of Geographic Names** (<https://www.getty.edu/research/tools/vocabularies/tgn/>)
- **The Alexandria Digital Library** (<http://adl-gazetteer.geog.ucsb.edu/>)
- Yahoo's **GeoPlanet** (https://en.wikipedia.org/wiki/Yahoo!_GeoPlanet)
- **Geonames** (<http://www.geonames.org/>)
- The joint effort of the New York Public Library (NYPL) and the Library of Congress (LoC) to create **an historical gazetteer** (<http://loc.gazetteer.us/>)
- The NYPL's recently announced **Space/Time Directory** (<http://spacetime.nypl.org/>)
- **NaturalEarth** (<http://www.naturalearthdata.com/>)

- **Quattroshapes** (<http://quattroshapes.com/>)

We use **Quattroshapes** (<http://quattroshapes.com/>) as the foundation for our data set, as it has most of the stuff, in most of the places, most of the time.

From there we are merging in relevant additions (geometries and metadata alike, as the various licenses permit) from **Natural Earth** (<http://www.naturalearthdata.com/>), **GeoPlanet** (https://en.wikipedia.org/wiki/Yahoo!_GeoPlanet), **GeoNames** (<http://www.geonames.org/>) and the **Zetashapes** (<http://zetashapes.com/>) project.

People who've been around this problem for as long as we have may be thinking: Doesn't this mean that the coverage for certain place types (many neighbourhoods around the world and in some countries even localities) will be lacking to start with? The answer is: Yes. One of our immediate goals is to identify which places in the GeoPlanet dataset lack both a concordance with and a corresponding record to Quattroshapes. Once we have that list we can import the names and hierarchies for those places from GeoPlanet at which point the coverage should improve immediately. Many of those places will lack coordinate data at the time of import but we believe that is a problem that can be tackled in time.

Ours is not the first effort to create an open and comprehensive data set of the world. We think that each one of the projects mentioned above offers their own unique contributions and we hope that by “smushing” them all together we can start to create something greater than the sum of its parts.

Also, all the data is made available under a **Creative Commons Zero** (<https://creativecommons.org/choose/zero/>) designation.

We make use of a number of open data sources, some of whom do require attribution. We have listed them **over here** (<https://github.com/mapzen/whosonfirst-data#sources>).

The long version



An interesting way to think about a gazetteer is to consider it as the space where debate about a place is managed but not decided.

Long version is long. You might want to get a cup of coffee or maybe a drink if you've been thinking about this sort of thing for as long as we have (or maybe longer).

What is a “gazetteer” anyway?

Like we said before, a gazetteer is a big list of places, each with a stable identifier and some number of descriptive properties about that location. An interesting way to think about a gazetteer is to consider it as the space where debate about a place is *managed* but not decided.

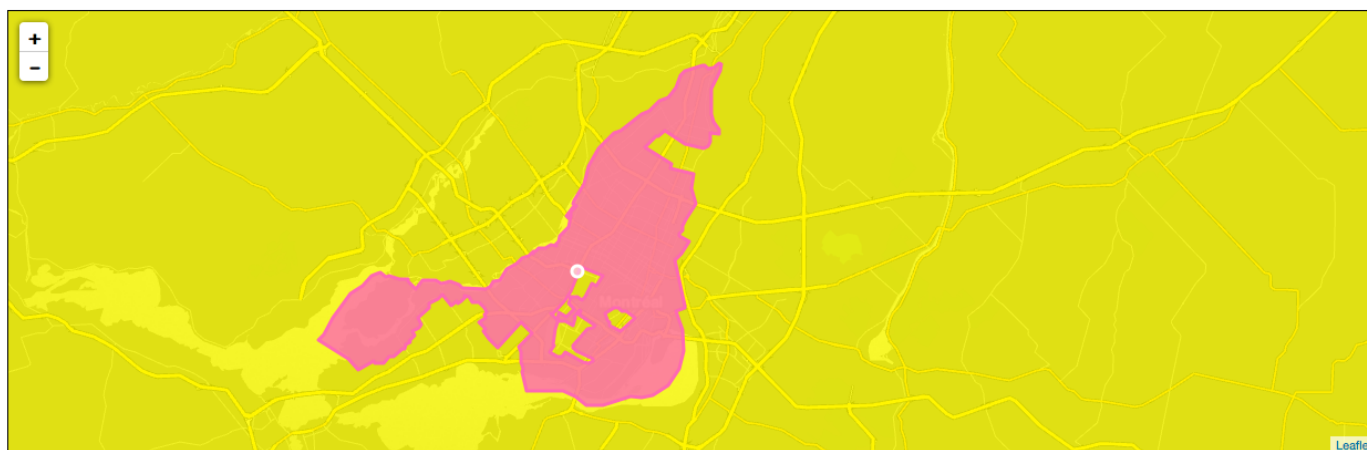
The simplest and friendliest expression of this idea is to consider the many different ways that people spell the name for the same physical location. Sometimes two people who agree on how to spell a name can find themselves at odds when one of them spells it incorrectly. Consider how finicky geocoders can still be about the manner in which a query is formatted and the hilarity that often ensues when a query is entered incorrectly. Now take this dynamic and multiply it by all the languages in the world.

Sometimes the easiest way to describe the purpose of a gazetteer is to say that it exists to make the “spelling mistake problem” go away. If every location has a simple (often numeric but more importantly *stable*) identifier then you and I can refer to the same place, in any number of different contexts, using that identifier without having to spend a lot of time accounting for linguistic differences or subtleties.

For example I might say “Montreal” (in English) while you might say “몬트리올” (in Korean) while someone else might say “Montréal” (in French) or “MTL” (in Conversational Shortcut) and so on. The gazetteer is meant to be a happy, welcoming and stable space where all those representations of the same place – or if you’re feeling frisky, the same consensual hallucination – can co-exist.

A more complicated expression of the idea that a gazetteer exists to manage debate is to remember that “place” is a hard problem because it is often a contested space at social, political and often very emotional levels. This is not a new problem. People have argued and sometimes fought over the meaning and the expression of place for as long as anyone can remember.

Montréal



Properties

```
{
  'u'geom:area': 'u'0.049595',
  'u'geom:bbox': 'u'-73.974351,45.409653,-73.474127,45.707578',
  'u'geom:latitude': 'u'45.52662',
  'u'geom:longitude': 'u'-73.652698',
  'u'gn:population': 'u'3268513',
  'u'iso:country': 'u'CA',
  'u'lbl:latitude': 'u'45.50884',
  'u'lbl:longitude': 'u'-73.58781',
  'u'name:chi_p': ['u'\u8499\u7279\u5229\u5c14'],
  'u'name:chi_v': ['u'\u8499\u7279\u5229\u723e'],
  'u'name:eng_p': ['u'Montreal'],
  'u'name:eng_s': ['u'Montreal',
    'u'City of Saints',
    'u'The Golden City',
    'u'City of a thousand bell towers',
    'u'La ville aux cent clochers',
    'u'The Belltower',
    'u'Sin City',
    'u'The Amsterdam of North America',
    'u'The City Light',
    'u'The Lamp',
    'u'The Big Island',
    'u'The Metropolis',
    'u'Frenchtown',
  ]
}
```

Montréal is a **locality** and its consensus geometry is from [quattroshapes](#)

Hierarchy

Ancestors that Montréal is parented by

- the country of [Canada](#)
- the region of [Quebec](#)
- the continent of [North America](#)

Other

- [See all the descendants of Montréal](#)
- [Reverse geocode](#) (45.52662, -73.652698)

{}

Even with the luxury of complete sentences and full paragraphs and entire books on the subject different writers and scholars *still* after all these years can't or don't or won't agree on the meaning of a given place. It would be ... unhelpful, at best, to believe that any list of key-value pairs, no matter how comprehensive, attempting to describe the nuance of a place will do any better at resolving long-standing disputes about the fundamental meaning and interpretation about that place.

First principles

The gazetteer starts from a series of first principles:

Mapzen has an opinion

It is important that Mapzen have an opinion not about any one place but rather about *the nature of place* itself. It is important for us to know and understand the boundaries of our project in order to know what the project is for and, critically, what the project is not.

Leave as many decisions as possible to the “edges”

The world is a complicated place and we would like the gazetteer to be a project that can support, or act as a scaffolding for, the sometimes contradictory opinions that people have about it. We aim to leave as much meaning or inference, as we can, about a place to individual users and applications. How this will manifest itself in concrete terms remains to be seen but this is a goal we have set for ourselves.

Portability

The canonical source for a place is a text file, specifically GeoJSON with a unique 64-bit numeric ID. This is because all computers speak “text files” and “numbers”. Text files can be inspected or updated in any old text editor. Text files can be printed. Numbers are fast and cheap for databases to index.

We use text files because our primary concern for the data is: Ease of use, robustness and portability *over time*. On measure, the benefits of plain old text files outweigh both the costs and in many cases the benefits of other formats.

Google's **Protocol Buffers** (<https://developers.google.com/protocol-buffers/>) for example are awesome but require that you install a whole lot of Google on your computer in order to use them. ESRI's **Shapefiles** (<https://en.wikipedia.org/wiki/Shapefile>) are equally awesome and their ubiquity and longevity is a testament to their utility but they too require bespoke applications for even the most trivial of updates.

Our primary concern for the data is: Ease of use, robustness and portability over time.

That does not mean that plain text or static files are necessarily the optimal choice for delivery or distribution. We will account for that on a case-by-case basis. If we need to pre-process all the data into a smaller and nimbler format for a specific use-case then we will, but you will always be able to access the data as simple text files.

GeoJSON

We use **GeoJSON** (<http://www.geojson.org>) as the primary exchange format for the gazetteer for two interconnected and complementary reasons:

- It is structured data with the least amount of markup *today*. If someone creates another markup language with even less scaffolding we might use that instead but for now GeoJSON is a good happy medium.
- There are lots of tools for working with GeoJSON and, importantly, for converting it into all the other formats that different people use.

Tell me more (the hard bits)



If you are immediately curious about the simple bits, where “simple” means names, geometries and minimal viable properties (and which are simple relative only to what follows) then you should **jump ahead**.

Concordances

Concordances are a good place to start since we’ve already mentioned other gazetteers. Put simply: We want to hold hands with as many other gazetteers – past present and future – as possible.

Concordances allow everyone to work on the things that are important to them while still being aware and acknowledging the work that others are doing and provide a mechanism for all that work to hold hands.

There is more than enough room for multiple gazetteers in the world and the point of a concordance is only to be able to say, for example, that “our Boston” is the same as “their Boston”. The details of either one of those records may be entirely different because of an individual organization’s focus or perspective. Multiple outlooks on the same subject are a good thing. Concordances allow everyone to work on the things that are important to them while still being aware and acknowledging the work that others are doing and provide a mechanism for all that work to hold hands.

Every WOF record contains a property called `wof:concordances` which is a dictionary of simple key/value pointers to the identifiers of the same in other databases. For example:

```
"wof:id": 101736545,  
"wof:concordances": {  
  "fct:id": "03c06bce-8f76-11e1-848f-cfd5bf3ef515",  
  "gn:id": "6077243",  
  "gp:id": "3534"  
}
```

As of this writing we have concordances for things from GeoNames (159, 359 of them), GeoPlanet (135, 399 of them), QuattroShapes (115, 550 of them), Factual (80, 973 of them), a variety of airport codes (ICAO, IATA, FAA and OurAirports), Wikipedia pages (just for airports right now) and even concordances for **Mapzen Border countries** (<https://mapzen.com/data/borders/>). Soon, we will have more.

Placetypes

For any hierarchy of place types we have identified three “classes” that any one of those place types can fall into. That doesn’t mean there can’t be others (classes or place types) only that these are the ones we’ve identified as a good place to start.

Common(c)

These are, well, common across *any* hierarchy for any place in Who’s On First.

This part is important: It means that at some point every single record shares at least one or more common ancestors (for example a country or a continent or occasionally just the Earth). That doesn't preclude very specific additions to the hierarchy for a given location only that those additions need to fit within a common hierarchy shared across all locations.

Common-optional (CO)

These are meant to be part of a common hierarchy but may not be present because they aren't relevant or because we don't have the data. Counties are a good example of this.

Optional (O)

These are the parts of a hierarchy specific, typically, to a country or region. For example, the many nested "departments" in France or Germany. The only rule is that an optional (O) place type has to fit somewhere inside the common (C) hierarchy.

So the minimum list of place types for a hierarchy applied globally looks like this:

- continent (C)
- country (C)
 - region (C)
 - "county" (CO)
 - locality (C)
 - neighbourhood (C)

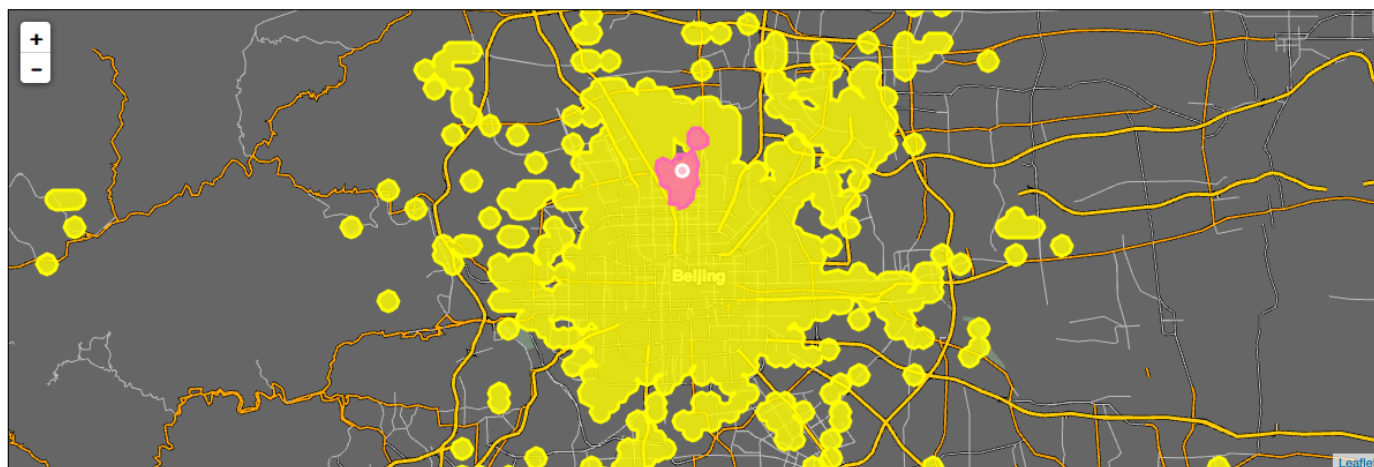
A more nuanced version might look like this:

- continent (C)
- empire (CO)
- country (C)
 - region (C)
 - "county" (CO)
 - "metro area" (CO)
 - locality (C)
 - macrohood (O)
 - neighbourhood (C)
 - microhood (O)
 - campus (CO)
 - building (CO)
 - address (CO)
 - venue (C)

Venues! Buildings!! **Microhoods** (<https://github.com/whosonfirst/whosonfirst-data/blob/master/meta/wof-microhood-latest.csv>)!!! Empires!!!!

So many new things but throughout you can still see a common skeleton. There is an **entire Github repository devoted to the topic of placetypes** (<https://github.com/whosonfirst/whosonfirst-placetypes>) including a discussion (and a canonical reference) about each of the place types described above.

Beijing International Convention Centre



Beijing International Convention Centre is a **neighbourhood** and its consensus geometry is from [quattroshapes](#)

Properties

```
{'geom:area': u'0.002119',
 'geom:bbox': u'116.350187,39.989747,116.412893,40.063899',
 'geom:latitude': u'40.02414',
 'geom:longitude': u'116.382464',
 'gn:id': u'',
 'iso:country': u'CH',
 'misc:gn_adm0_cc': u'CN',
 'misc:gn_id': u'0',
 'misc:gn_local': u'1816670',
```

Hierarchy

Ancestors that Beijing International Convention Centre is parented by

- the country of [China](#)
- the region of [Beijing](#)
- the locality of [Beijing](#)

Other

- [See all the descendants of Beijing International Convention Centre](#)
- [Reverse geocode](#) (40.02414, 116.382464)

Hierarchies

Hierarchies in Who's On First are represented as a list of dictionaries, where each item is a dictionary containing a full hierarchy. Like this:


```
"wof:hierarchy": [  
  {  
    "country_id": "85633147",  
    "region_id": "85683255",  
    "county_id": "102072387",  
    "locality_id": "101750223",  
    "neighbourhood_id": "85794581"  
  }  
]
```

This is because a location in Who's On First can have *multiple* divergent hierarchies. Consider place types like metropolitan areas (the “Bay Area” in and around San Francisco, “New York” inclusive of all five boroughs and even some parts of New Jersey and so on) which often contain multiple children of the same place type. Consider disputed territories. There is a whole separate document about the rationale for this decision and how we arrived at it but the short excerpted version is this:

Reasons why this suggestion is good:

- It is explicit
- It is easy to compare multiple hierarchies
- It doesn't require the user do a lot of mental arithmetic to construct the complete hierarchy or to support whatever “efficiencies” we dream up in the moment
- It is easier to change going forward (say before an “official” launch) than the alternatives

Reasons why this suggestion is, or might be, bad:

- If we support metropolitan areas, then many places (localities, neighbourhood, venues) may have multiple hierarchies where the only difference will (likely) be the county, leaving all the remaining ancestors in common
- File size, disk space and bandwidth - this is the corollary of the first point and akin to whitespace or coordinates with > 6 decimal points in GeoJSON files which can become burdensome quickly

Disputed areas

Although it is true that all neighbourhoods and many localities are “disputed” in a friendly-banter kind of way some places are genuinely more disputed than others because they involve two or more countries (and sometimes so-called **non-state actors** (https://en.wikipedia.org/wiki/Non-state_actor)), a real threat of violence and consequences that in no way resemble “friendly banter”.

For the time being we have explicitly assigned places we know to be disputed a place type of `disputed`. As of this writing disputed places are parented, in their hierarchies, by two or more countries. This approach does not reflect the subtleties of the facts on the ground in any, of these disputed places. On the other hand it does allow to “block out” areas that are contested and, as we’ve said above, to leave the decisions about how to reflect the dispute to individual applications.

Parent ids and “custody”

Even though a place can have multiple hierarchies we still assume that in most cases there is a single “authority” in place, on the ground. For example, the Golan Heights is disputed by both Syria and Israel, a fact reflected in the place hierarchy, but is still “parented” by Israel:

```
"wof:hierarchy": [  
  {  
    "continent_id": "102191569",  
    "country_id": "85632315",  
    "disputed_id": "85632221"  
  },  
  {  
    "continent_id": "102191569",  
    "country_id": "85632413",  
    "disputed_id": "85632221"  
  }  
],  
"wof:id": "85632221",  
"wof:name": "Golan Heights",  
"wof:parent_id": "85632315",
```

In a few isolated cases we really don’t have a good answer for who the parent is or if we’re just not sure because it’s early days and are still vetting the data we will assign a value of `-1` for the parent record.

Occasionally we will assign a value of `-2`. This should be interpreted as “shrug: the world is a complicated place”. For example, **the Baykonur Cosmodrome is a little donut hole of Russia** (<https://github.com/mapzen/whosonfirst-data/issues/16>) surrounded by the nation of Kazakhstan.

Superseded / Superseded by

One of the large and difficult philosophical questions that working with geography raises is this: *How do you decide when something has simply been updated versus when has something fundamentally changed?*

This is not a “geo” question per se, only that “geo” has a habit of poking it in the eye. For example Poland, France and Germany spent the better part of a hundred years claiming and reclaiming (and then sometimes reclaiming again) the same territory. Those borders, and the periods during which they existed, are important contextual information for lots of applications beyond just maps. Consider works of art created in those parts of Poland that were German at the time of their creation. How do you create stable identifiers for a location that has changed, in concrete and meaningful ways, over time?

One of the large and difficult philosophical questions that working with geography raises is this: How do you decide when something has simply been updated versus when has something fundamentally changed?

A little closer to home, in New York City, everyone's favourite *that-is-so-not-a-neighbourhood* is called: “BoCoCa”. BoCoCa is short for Boerum Hill, Cobble Hill and Carroll Gardens, three adjacent neighbourhoods south of downtown Brooklyn. BoCoCa is neither a name that anyone seems to like, nor is it a place type that most people recognize as a neighbourhood. On the other hand it is a neighbourhood (and an identifier) that has made its way into lots of other maps and datasets. Whatever we may think about it BoCoCa has “existed”.

So in Who's On First we've changed its place type and made BoCoCa a “macrohood” that parents the three neighbourhoods from which its name is derived.

A location's place type is a pretty fundamental property and it's easy to imagine that a third-party application might use and store that metadata for important application-specific purposes. Which is to say: We don't need to have an opinion about what or why an application is doing something with the metadata associated with a location and if we start by saying that WOF ID 85892915 is a neighbourhood (which it was when we imported the data from Quattroshapes) *we probably shouldn't just change it on a whim.*

We also don't think BoCoCa is a neighbourhood, though. We absolutely have an opinion about that. At one time BoCoCa might have been a neighbourhood but in our world it no longer is. The way that we reconcile problems like this is by creating a new record with a new place type (BoCoCa the macrohood) and by updating each record to indicate that one supersedes the other.

For example, the record for BoCoCa the neighbourhood looks like this:

```
"wof:superseded_by": [102147495],  
"wof:supersedes": [],
```

While the record for BoCoCa the macrohood looks like this:

```
"wof:superseded_by": [],  
"wof:supersedes": [85892915]
```

It is left up to individual applications to decide how or whether to treat records that have been superseded differently. A search engine, for instance, may choose to weight a superseded place differently or exclude it altogether.

Breaches

Every record has a `wof:breaches` list attached to its properties. Depending on when you read this blog post, most or all of those lists may still be empty. A “breach” occurs when the geometry of a given location is intersected by the geometry of another location of the same place type.

The purpose of these lists is to signal to users of the Who's On First data that there is either an error in the data (most country records shouldn't breach any of their neighbours) or that there is a legitimate difference of opinion about the boundaries of a place (neighbourhoods, for example).

Like many signals its precise meaning and significance and how it should be handled is left up to individual applications.

Tell me more (the simple bits)



Remember, what follows is the “simple” stuff...

Names

All place names were originally derived from the Quattroshapes and Natural Earth datasets. Overall though the GeoPlanet (GP) dataset is better for names in multiple languages, and for colloquial place names. GP defines two properties for a name:

1. An **ISO 639-3** (https://en.wikipedia.org/wiki/ISO_639-3) language code
2. A name “type”, which is a well-known list of descriptors as defined by the nice GP people:

The Name_Type field is a single letter code that describes the alias as follows:

- P is a preferred English name
- Q is a preferred name (in other languages)
- V is a well-known (but unofficial) variant for the place (e.g. “New York City” for New York)
- S is either a synonym or a colloquial name for the place (e.g. “Big Apple” for New York), or a version of the name which is stripped of accent characters.
- A is an abbreviation or code for the place (e.g. “NYC” for New York)

GP also distinguishes between a `name` and an `alias` so in their world you end up with something like:

```
Name: Montréal
Language: FRE
Alias (ENG_P): Montreal
Alias (KOR_Q): 몬트리올
```

GP does not however account for the fact that **some countries**

(<https://www.youtube.com/watch?v=OHzMTSK1V4o>) have multiple languages. With all that in mind, we decided that:

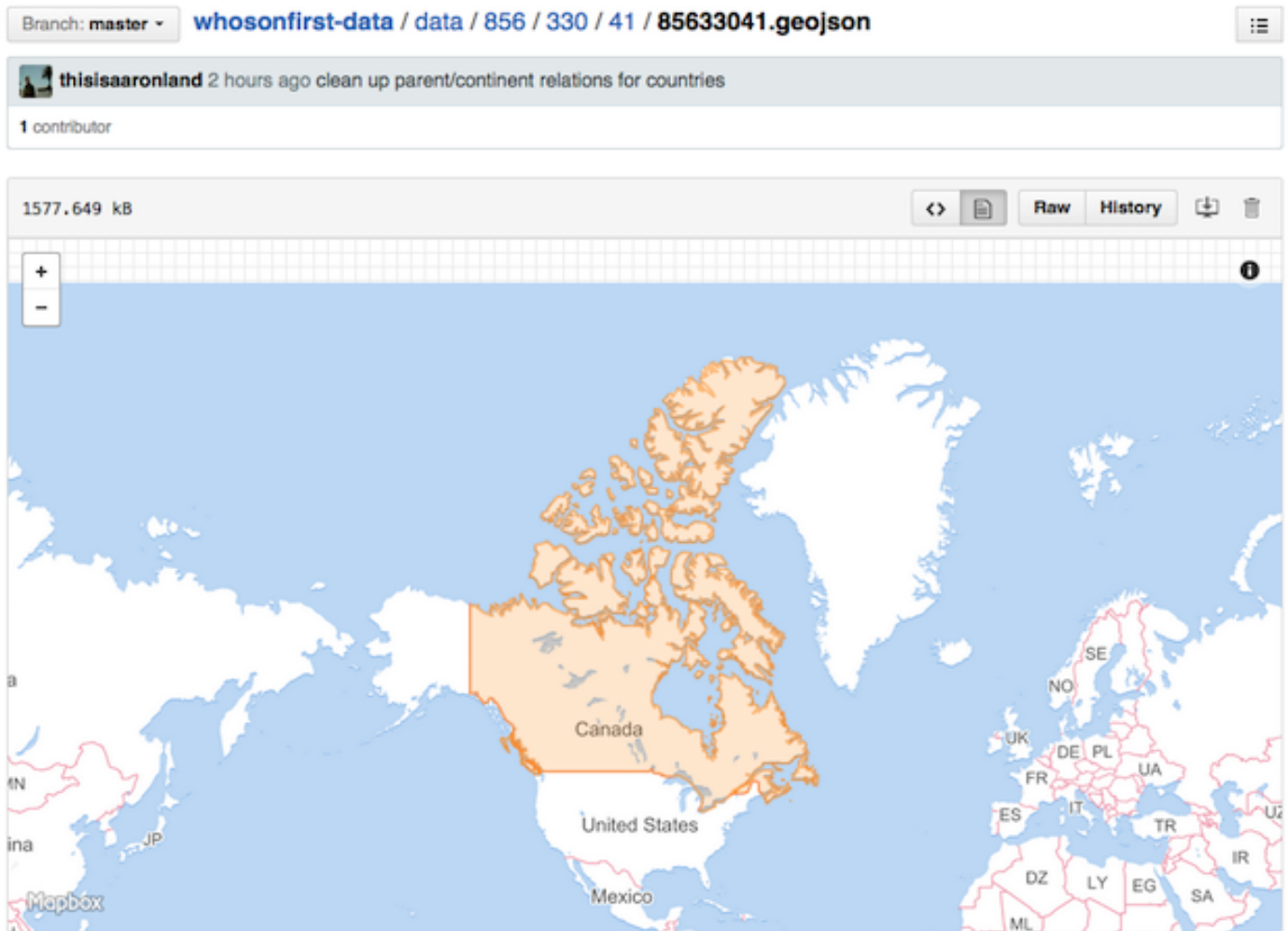
- We should support multiple languages for a place and label placement
- We should just use `p` for a preferred name, regardless of language
- We should use a `name` namespace for names because it is explicit

For example:

```
{
  "wof:lang": ["eng", "fre"],
  "name:eng_p": "Montreal",
  "name:eng_a": "YMQ",
  "name:fre_p": "Montréal",
  "name:kor_p": "몬트리올",
}
```

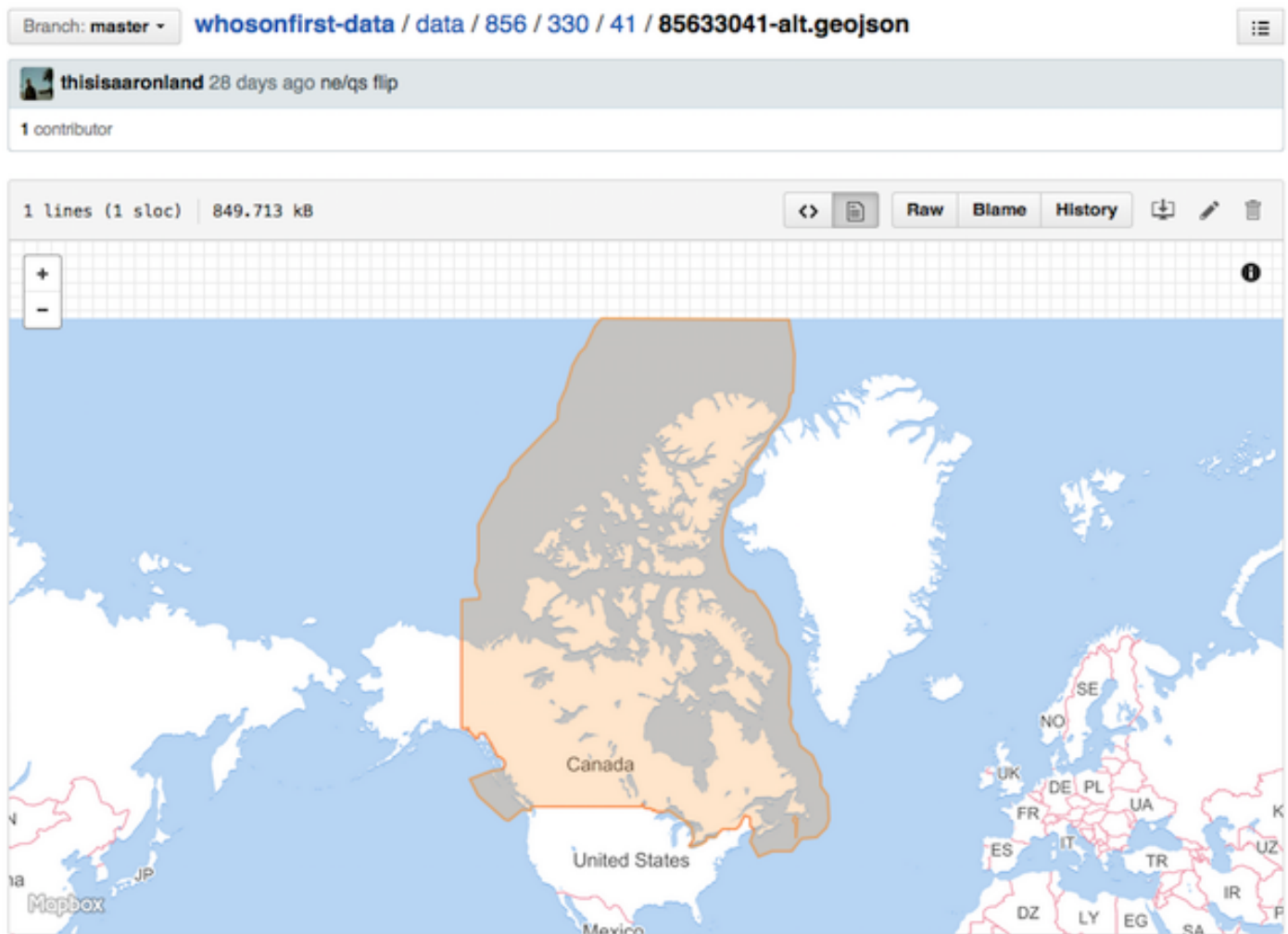
Geometries

The “consensus” geometry



Every place will have a single “consensus” geometry associated with it. The definition of “consensus” remains to be determined. Also, the use of the word “consensus” is understood to be problematic. It *will* be deprecated and replaced with a more accurate term.

All the “other” geometries



Every place will also have an “alternate” file containing multiple named geometries. These might include contested geographies or simplified geometries or geometries that have been optimized for specific use-cases, like geocoding.

The point is this: ALL THE GEOMETRIES.

The source for the consensus (sic) geometry and all the alternate geometries are included in every record. For example:

```
{
  "src:geom": "zetashapes",
  "src:geom_alt": ["quattroshapes", "naturalearth"]
}
```

Centroids

Any given record can have multiple centroids. It is worth pointing out that “multiple centroids” can be read as a bit of an oxymoron. We use the term “centroid” to denote a specific area of focus for a geometry. Different centroids are denoted by prefixes, indicating their use. For example:

- `geom:latitude` and `geom:longitude` – which are the actual center of the consensus polygon and calculated using the magic of mathematics.
- `lbl:latitude` and `lbl:longitude` – which are used to indicate the optimal location for label placement. For example, the polygon for San Francisco includes the Farallon Islands which means that the “center” of that polygon is actually in the Pacific Ocean and not an optimal place to label the city.
- `nav:latitude` and `nav:longitude` – which are used to indicate where, specifically, a routing engine should direct you. For example, the entrance to the ER at a hospital rather the loading bay for delivery trucks arriving at the same location.

Minimal viable properties

The design model for the **Flickr API** (<http://www.flickr.com/services/api>) was: *What is the one thing you should always be able to do with any photo-related API response?*

For Flickr the answer was the “**standard photo response** (<https://code.flickr.net/2008/08/19/standard-photos-response-apis-for-civilized-age/>)” or: *You should always be able to load/build the URL for a photo with a link back to its page on Flickr’s website.*

In Mapzen’s case the answer to a similar-minded question might be: *You should always be able to view API responses on a map.*

For example it should be possible to take anything that comes back from **Pelias** (<http://pelias.mapzen.com/>) (or any other API) and simply hand it off to Leaflet as a GeoJSON layer and see something on a map.

With that in mind a “minimal viable properties” dictionary might look like this:

```
{
  "wof:id": 85922583,
  "wof:name": "San Francisco",
  "wof:fullname": "San Francisco, California US",
  "wof:placetype": "locality",
  "wof:parent_id": 85688637,
  "wof:quality": 9,
  "wof:score": 100
}
```

A couple things to note about this example:

- When we say “properties” we’re talking about the metadata associated with that place rather than its geometry.
- The `wof:score` property and its semantics should be considered a place-holder for some equivalent search ranking to be determined by the Pelias team. Likewise, the `wof:quality` property and its semantics should also be considered a place-holder for some equivalent quality/confidence ranking to be determined by the Data team.

Future magic (making it)

What follows are a list of properties that aren’t currently supported or are still so “early days” that the easiest thing is to consider them unsupported. They are included here because they absolutely *should and will* be supported, in time.

Ranking(s)

Quality

How good or confident do we feel about the data included in a given record.

Coverage

Where “coverage” is meant to indicate the number of attributes that have been recorded about a place. We might have excellent names and alternate names for a place but little else in the way of metadata (population, elevation, etc.)

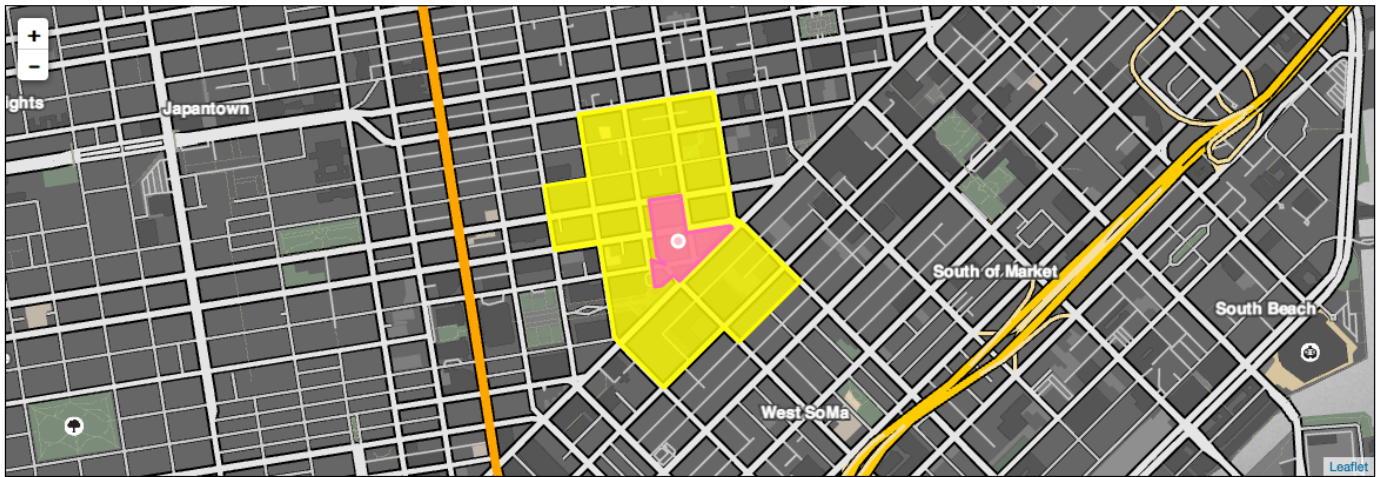
Dates

What are the dates that a place was founded or, in some cases, ceased to exist? This becomes especially relevant in a dataset where every record might be superseded or supersede another.

Dates are actually a pretty complex space so we are going to start with the simplest thing and then gradually grow it to account for the realities of life and of history. The Library of Congress' work around an **Extended DateTime Format (EDTF)** (<http://loc.gov/standards/datetime/>) is a good place to start poking around if you're curious about this sort of thing.

But wait... where is this data you speak of?

The Yo



Properties

```
{u'geom:area': u'4e-06',
 u'geom:bbox': u'-122.413719,37.780369,-122.41066,37.78302',
 u'geom:latitude': u'37.781722',
 u'geom:longitude': u'-122.41265',
 u'iso:country': u'US',
 u'name:eng_p': [u'The Yo'],
 u'src:geom': u'minitenders',
 u'src:geom_alt': [],
 u'wof:belongsto': [u'85688637',
 u'85633793',
 u'85922583',
 u'102087579',
 u'85865903'],
 u'wof:breaches': [],
```

The Yo is a **microhood** and its consensus geometry is from [minitenders](#)

Hierarchy

Ancestors that The Yo is parented by

- the county of [San Francisco](#)
- the country of [United States](#)
- the region of [California](#)
- the neighbourhood of [Tenderloin](#)
- the locality of [San Francisco](#)

Other

- [See all the descendants of The Yo](#)
- [Reverse geocode](#) (37.781722, -122.41265)

{}

First – and this is Very Very Very Important – please understand that Who's On First is a work in progress and that means a few things:

- Some (maybe even a lot) of the data will be wrong.
- Some things are missing. Some things are missing in a **known unknown** (<https://github.com/TileStache/TileStache/blob/fb60b32439ebf108f983eaa4556627b07848d7ad/TileStache/Core.py#L772-L821>) kind of way in which case they'll be addressed

shortly. Some things may still be missing in an **unknown unknown** (https://en.wikipedia.org/wiki/There_are_known_knowns) kind of way in which case they'll be addressed as the errors become apparent.

- Some (probably most) of the data will change in some way, if only to account for #1.
- We have not formalized or finalized the tools for updating all the ancestors or dependencies of a record when that record is updated. This means that in the short-term it is possible there will be inconsistencies between a record and its relations. We'll get there.

The purpose of releasing the data now is not to sound the trumpets and herald a new dawn of perfect data but rather to give substance to everything we've been talking about and to have a meaningful dataset with which to prove or disprove our assumptions and to work through the practicalities of working with that data.

A lot of work is what's next.

If you don't have the time or the temperament (personally or institutionally) to deal with a little bit of on-going **bad craziness**

(<http://www.ralphsteadmanprints.com/images/00art/gonzo/02badcrazy.jpg>) as we work through the issues then digging in to the data now is probably premature. We intend to continue working in public and discussing the project openly so keep an eye on the blog and we'll let you know as things improve.

As of this writing the raw Who's On First GeoJSON data files exist in two places: A public AWS S3 endpoint and as a GitHub repository with lots of tiny little files. Their URLs respectively are:

- <https://s3.amazonaws.com/whosonfirst.mapzen.com/data/>
(<https://s3.amazonaws.com/whosonfirst.mapzen.com/data/>)
- <https://github.com/whosonfirst/whosonfirst-data/>
(<https://github.com/whosonfirst/whosonfirst-data/>)

Note: The S3 link above is not meant to be a "browseable" thing so much as an endpoint that people know how to work with S3 can start to do stuff with. If those words sound like gibberish to you then you should not click on the S3 link and just have a look at the GitHub link instead.

There is not currently a publicly available tool for browsing the data. We have an internal "spelunker" that we plan to open-source (along with a number of related libraries for working with Who's On First data) but that is not going to happen today.

The GitHub repository also contains a number of “meta” files, located in the cleverly-named **meta directory** (<https://github.com/whosonfirst/whosonfirst-data/tree/master/meta>). These are mostly CSV files with some minimum amount of data for each record of a given place type. Like everything else the meta files are a work in progress but are meant to provide a minimal ability to introspect the data without the need to load everything into a database.

Did you say... “venues” ?

We did. We aren't releasing any venue data under the Who's On First banner, yet, but we are working on it. Venues are very much a part of Who's On First but where they are not complicated they are simply numerous. Often they are both. So, simple things first. Harder things, like venues, will follow.

A word about Git (and GitHub)

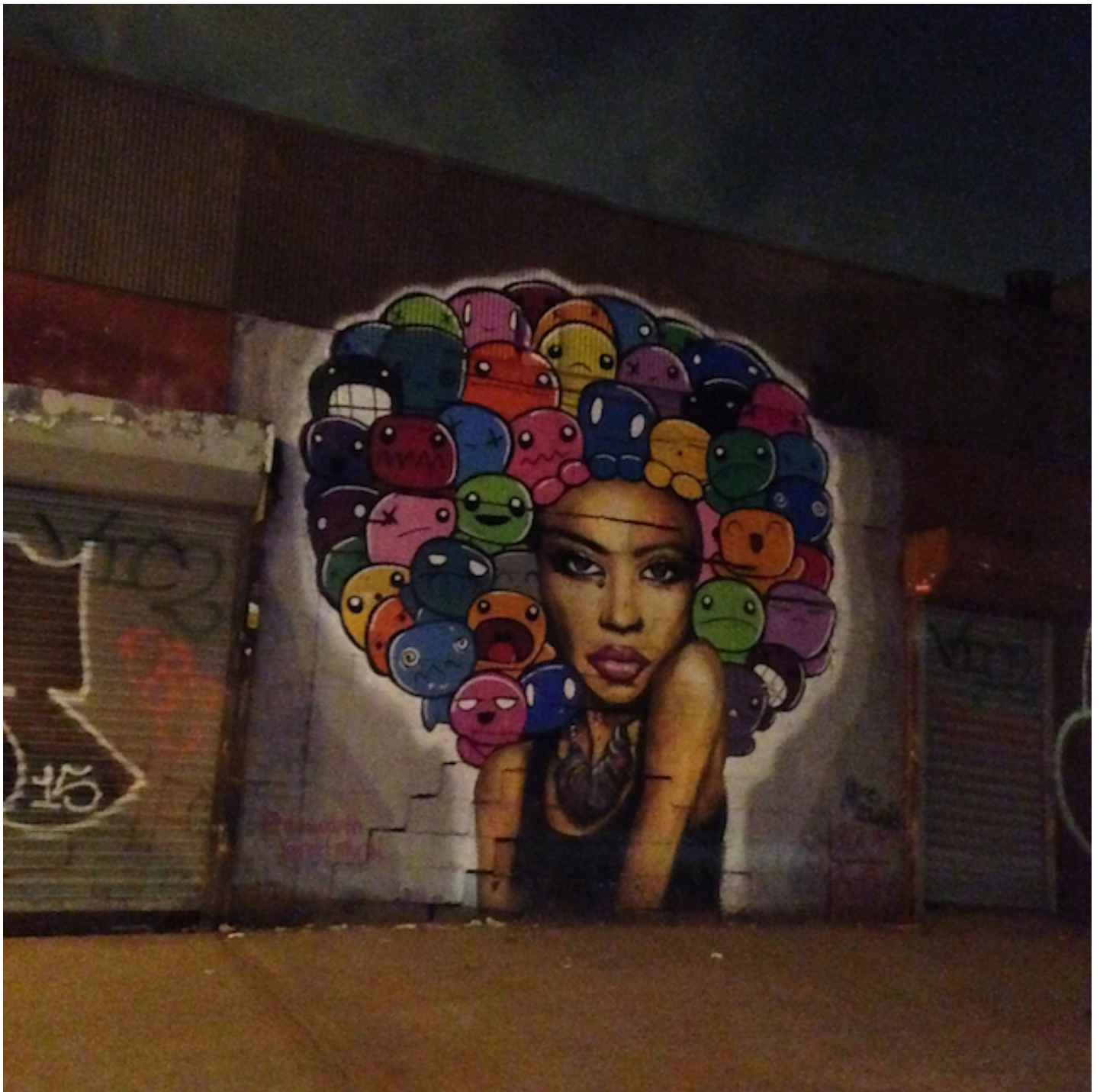
Don't get too attached to working with or managing Who's On First data in GitHub (or Git in general). We haven't quite figured out what the best way of both distributing the Who's On First data or of accepting corrections and suggestions from the community is yet.

Even though the nice people at GitHub continue to do excellent work at making Git easier for a broader population to use, the reality remains that Git is a significant barrier to participation for many people. Absent a more formal decision about an alternative, GitHub at least allows us to point in the general direction of:

- An open and readily distributed dataset that people can download and work with
- A way for people to contribute corrections (and general nuance) about a place
- A way for us to be able to do everything above while still assuring us a measure of authority around the assertions we make about the data
- Also a way for us to think about how and where we store an audit trail (of sorts) for updates to a place

So again: Don't get overly attached to the Git parts of the Who's On First data. It is meant to indicate the idea and not necessarily the details.

What is next?



A lot of work is what's next.

More specifically what's next is to start releasing the tools and libraries we've written internally for processing and managing Who's On First data, releasing a version of the internal "spelunker" web application that we use to poke around the data and perform sanity-checking, building prototype services on top of all this data, finishing and in some cases starting documentation for all of the above and **fixing all the bugs** (<https://github.com/whosonfirst/whosonfirst-data/issues>).

Enjoy!

all images by Aaron Cope

· 18 August 2015 ·



Aaron Straup Cope

Aaron is happiest in the presence of olive oil.



Nathaniel Vaughn Kelso

Map geek, cartographer, and data omnivore. Nathaniel leads the data team at Mapzen and vacations @nullisland.

© 2017 Mapzen

Summer Fridays Open Office Hours

Meet the Mapzen team in our NYC office and learn how to use our free and open services for your geo needs. We will have people from our **search** (<https://github.com/pelias>), **navigation** (<https://mapzen.com/projects/valhalla>), **cartographic** (<https://mapzen.com/projects/tangram>) and **transit** (<https://transit.land/>) teams to answer questions, share the roadmap, and help you get started making your own maps.

Friday, August 28 at 12:00PM

30 W 2th St, 7th Floor, NYC

<https://sasf-open.splashthat.com/> (<https://sasf-open.splashthat.com/>)

At Mapzen we believe that open source and open data can usher in a new era of mapping. We are looking to shift an ecosystem and invite you be part of it. In other words, we go open big beach ball style.



So we'll be ready with snacks and maybe a **gigantic summer party beach ball** (<https://www.vat19.com/item/gigantic-beach-ball>) to introduce you to our tools and answer your mapping questions. Our doors, our office and our work are always open. Come join us! **RSVP now** (<https://sasf-open.splashthat.com/>).

*image and beachball via **Vat19** (<https://www.vat19.com/item/gigantic-beach-ball>)*

· 19 August 2015 ·



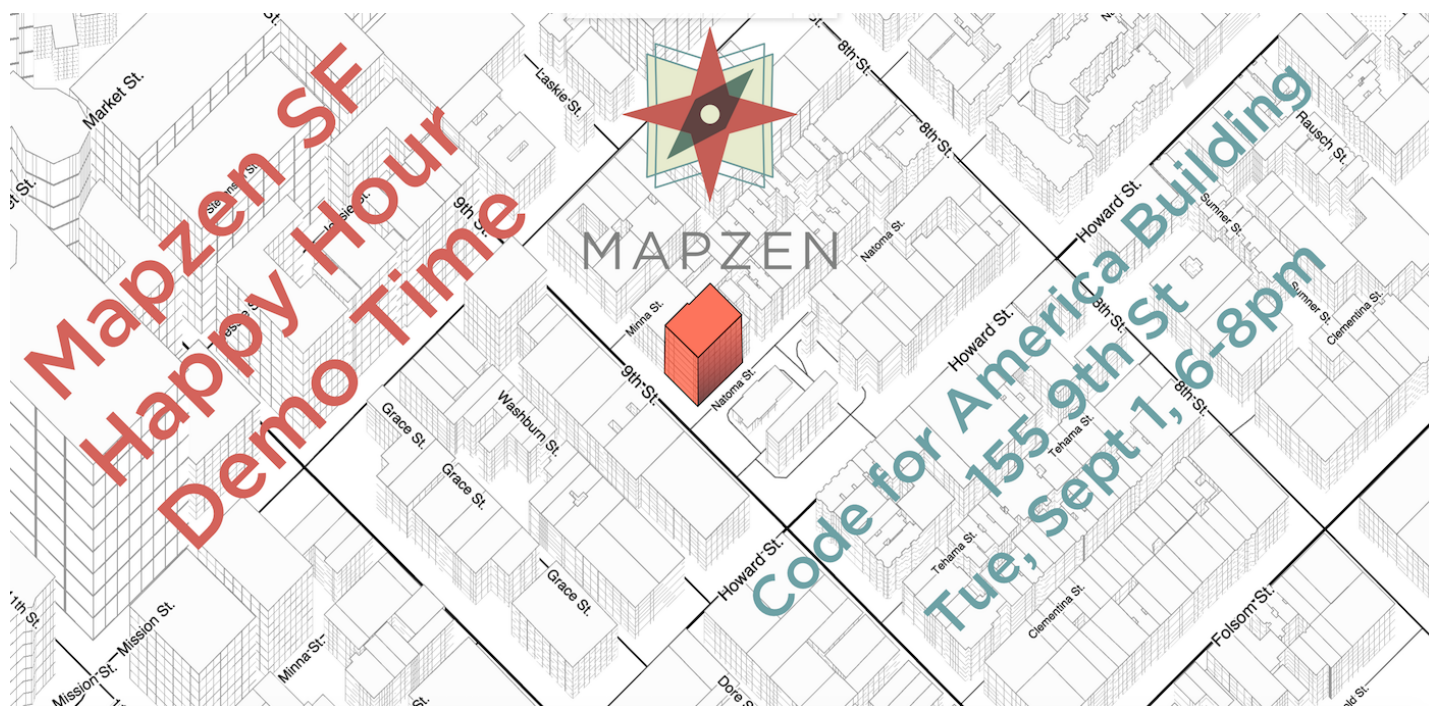
Alyssa Wright

Alyssa Wright does community where the phone and meeting are her superpowers of choice.

© 2017 Mapzen

Mapzen SF Happy Hour Demo Time

Not on the East Coast and sad you can't be at **Mapzen's NYC Summer Fridays Office Hours** (<https://mapzen.com/blog/office-hours>)? Never fear! Mapzen has you covered on both sides of the continent – on Tuesday, September 1, join us for the first ever **Mapzen SF Happy Hour Demo Time** (<https://www.eventbrite.com/e/mapzen-sf-happy-hour-demo-time-tickets-18300496277>)!



(<https://www.eventbrite.com/e/mapzen-sf-happy-hour-demo-time-tickets-18300496277>)

Join us at the CfA office at 6 for pizza and beer and other delicious solids and liquids while you learn about the open source software and data that we use to make the future of maps. We'll have people on hand to demo and talk about:

- WebGL maps (**Tangram** (<https://mapzen.com/projects/tangram>)!)
- routing (**Valhalla** (<https://mapzen.com/projects/valhalla>)!)
- vector tiles (**GeoJSON! TopoJSON! MVT!** (<https://mapzen.com/projects/vector-tiles>)
- search (**Pelias** (<https://mapzen.com/pelias>)!)
- gazetter (**Who's on First** (<https://mapzen.com/blog/who-s-on-first>)!)
- transit (**Transitland** (<https://transit.land/>)!)
- mobile! Android! labels! YAML! API keys!

We'll even have a few special guests flying in from our New York City office! (If you **sign up for an API key** (<https://mapzen.com/developers/>), Randy & Mike & Alyssa will wear white after Labor Day.)

RSVP now (<https://www.eventbrite.com/e/mapzen-sf-happy-hour-demo-time-tickets-18300496277>), and bring yourself and your geofriends and your geoquestions!

· 24 August 2015 ·



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

Find better bicycle routes with new Valhalla options

routing (/tag/routing)

Valhalla, Mapzen's free and open source routing and navigation engine, has been updated with an assortment of bicycle navigation enhancements. You can now influence the path of the bicycle route and the estimated travel time by including optional costing settings for the type of bicycle, your speed, or whether you want to ride on streets with traffic.

By specifying the bicycle type in a query to the **routing service API** (<https://github.com/valhalla/valhalla-docs/blob/master/api-reference.md>), Valhalla can suggest more appropriate routes and accurate times for you. While some kinds of bicycles are designed only for paved surfaces, others also work well on gravel and dirt paths, and Valhalla can route and estimate accordingly. Each bicycle type has a default cycling speed, which is intended to be what an average rider can comfortably maintain over the length of the route. You may ride faster or slower than the defaults, so you can now override the speed with your own value.

For example, if you are riding a road bike (a lightweight bicycle with narrow tires), you want your route only to follow paved surfaces. Perhaps you also want to take a slower-paced trip and minimize riding on roads where you must travel with cars.

The JSON for this query might look something like this:

```
valhalla.mapzen.com/route?json={"locations":[{"lat":34.413,"lon":-119.859}, {"lat":34.412,"lon":-119.846}], "costing":"bicycle", "costing_options":{"bicycle":{"bicycle_type":"Road", "cycling_speed":17, "use_roads":0.1}}} &api_key=valhalla-xxxxx
```

This particular query builds a bicycle route to the library at the University of California, Santa Barbara. It includes parameters for riding a `Road` bicycle at a `cycling_speed` of 17 kilometers per hour. The `use_roads` value is a range from 0 to 1, indicating an increasing tolerance for riding with vehicle traffic. By specifying a value closer to 0, though, your query

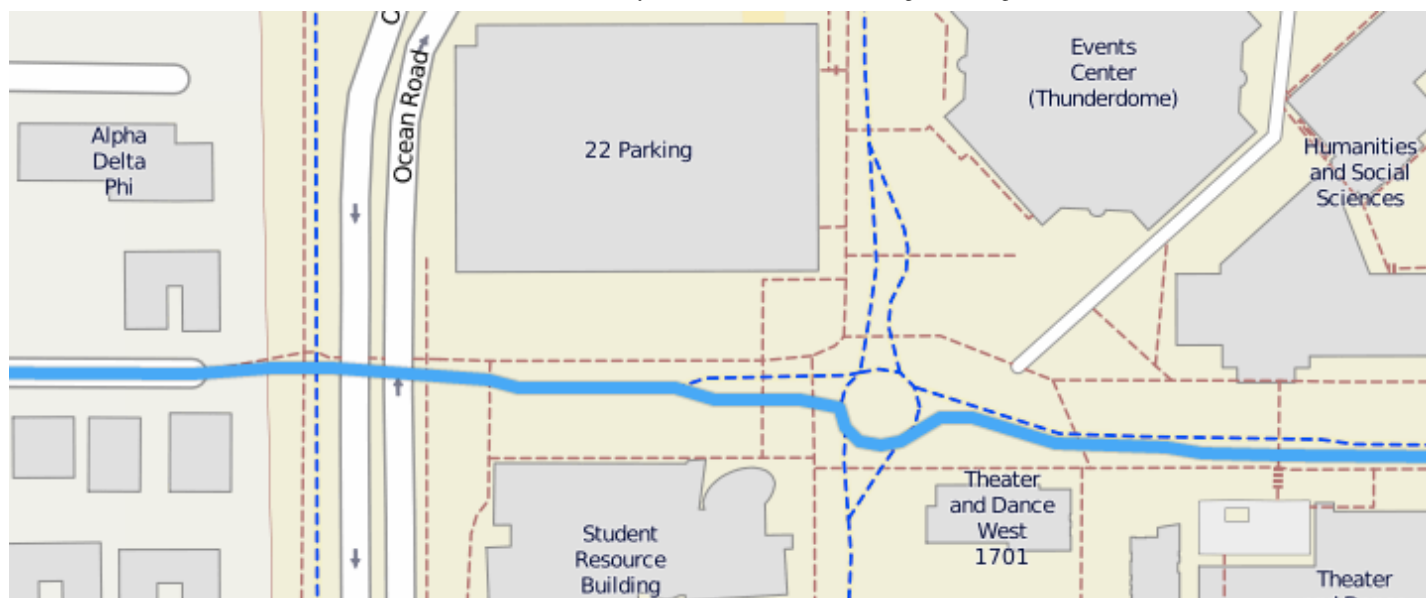
shows a preference for designated bicycle paths, when available. (Note that you would need to substitute **your own API key** (<https://mapzen.com/developers/>) at the end to perform the query.)

To visualize your trip, which is a little over a kilometer in length, you can draw the route line over an **OpenCycleMap** (<http://www.opencyclemap.org/>) basemap, which is built from OpenStreetMap data to emphasize cycling. On OpenCycleMap, cycle paths are shown with dashed blue lines, and footpaths are the dashed brown lines.



Here is an overview map of your bike route (shown in solid blue) from the neighborhood to the campus library to the east. A fun fact is that by car, traveling between these same points is three times the bicycle distance because vehicles cannot access such a direct route.

The Valhalla response provides distances and estimated times en route for the entire trip, as well as details for the component legs. Each maneuver has turn-by-turn instructions, including for the bicycle-only roundabouts you will pass through on this trip.



Here is a closer look at some of the maneuvers on your route. The path changes to bicycle-only access when you enter the campus, and then you travel through several roundabouts.

If you choose not to set any bicycle costing options, the route is tuned for road bicycles and the kinds of roads they can access, with a slight preference for using **cycleways** (<http://wiki.openstreetmap.org/wiki/Key:cycleway>) or roads with bicycle lanes.

Keep in mind that Valhalla bicycle routes do not currently consider elevation factors such as hills and steepness, but this is coming soon.

To get started with Valhalla, sign up for your free API key at <https://mapzen.com/developers> (<https://mapzen.com/developers>) and **check out the documentation** (<https://github.com/valhalla/valhalla-docs>).

*Credits for this post's screenshots: OpenCycleMap © **Thunderforest** (<http://www.thunderforest.com/>), data © **OpenStreetMap contributors** (<http://www.openstreetmap.org/copyright>).*

· 26 August 2015 ·



Rhonda Glennon

Rhonda is Mapzen's technical publications manager and writes about maps and developer tools.

© 2017 Mapzen

Dynamic Costing via Sif

routing (/tag/routing)

SIF – Dynamic Costing within Valhalla

Two core components of the Valhalla open source routing engine are **Thor** and **Sif**. These 2 companions (in Norse mythology Thor and Sif are husband and wife) form the basis of Valhalla's path generation algorithm. Thor contains the path computation algorithms and traverses the routing tiles, while Sif performs **costing** that is central to forming the best path. Rather than baking costs into the routing graph data, Valhalla uses dynamic, run-time costing to generate costs based on a rich set of attributes stored in the routing graph tiles. This allows run-time generation of different types of routes (or routes with different characteristics) simply by using different costing methods and options within those methods.

Valhalla uses dynamic, run-time costing when computing route paths and can consider much more than strict time or distance.

Path Costing Introduction

Routing from one location to another is solved by a class of algorithms known as **shortest path algorithms**. This is somewhat misleading, as often one is interested in a route that is shortest time or one that makes fewer turns. A better term for shortest path algorithms is **least cost algorithms** - this properly indicates that the method is minimizing cost, be it distance, time, or some other metric.

Naive assignment of cost to edges of the routing graph will lead to poor routing solutions. Simple costing based solely on distance or on time (based solely on speed) can lead to poor route paths with excessive turns and stops. Considerations such as turn types, classifications of roads at intersections along the route, road surface type, elevation change, road curvature, and a host of other considerations can be important. It is also important to note that different costing considerations are needed for bicycle routing than pedestrian routing or automobile routing.

Dynamic Costing

Valhalla uses dynamic, run-time costing when computing route paths and can consider much more than strict time or distance. Different route types can be computed from a single set of route data tiles. There is no need to configure data each time a new **routing profile** is needed. Simply change the costing methods or apply different options to existing costing methods, the data stays the same.

Costing Interface

Costing methods have access to all attributes of an edge (road section between 2 intersections) to form the cost along the edge and when transitioning between edges. Within Sif, costing methods are created by deriving a class from the base dynamic costing class or one of the existing costing classes. Each costing method must override 3 different methods to create the unique costing logic:

```
virtual bool Allowed(const baldr::NodeInfo* node) const = 0;
```

Checks if access is allowed for the provided node. For example, node access can be restricted for specific modes of travel if bollards are present.

```
virtual bool Allowed(const baldr::DirectedEdge* edge, const EdgeLabel& pred) const = 0;
```

Checks if access is allowed for the provided directed edge based on the prior edge along the path. This is generally based on mode of travel and the access modes allowed on the edge. It can also be used to prohibit turns where turn restrictions exist, prohibit Uturns, and to prohibit entering roads that do not have through paths.

```
virtual Cost EdgeCost(const baldr::DirectedEdge* edge, const uint32_t density) const = 0;
```

This method gets the cost to traverse the specified directed edge. Cost includes a path cost along with the actual time (seconds) to traverse the edge. Path costs are generally time or distance and can include artificial cost penalties to avoid roads/edges with specific attributes. By returning the actual elapsed time in seconds the costing method can be applied to time dependent and schedule based routing (e.g. transit).

Costing methods can also compute **edge transition costs**, sometimes called turn costs. These costs are applied at the node/intersection when transitioning from one edge to another. A fourth costing method can be defined in the costing class to account for this:

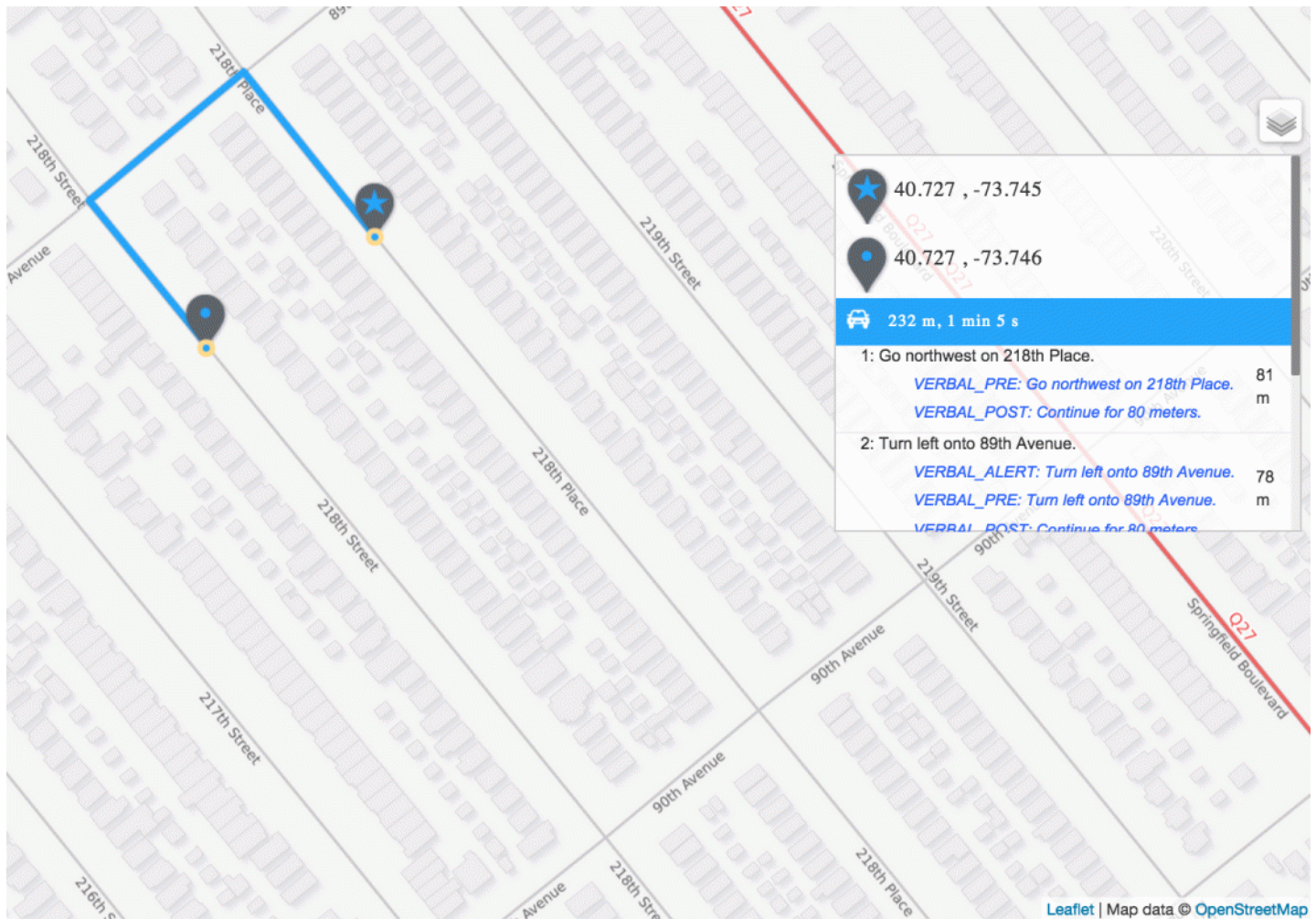
```
virtual Cost TransitionCost(const baldr::DirectedEdge* edge,  
                           const baldr::NodeInfo* node,  
                           const EdgeLabel& pred) const;
```

Edge transition costs generally consider 3 things:

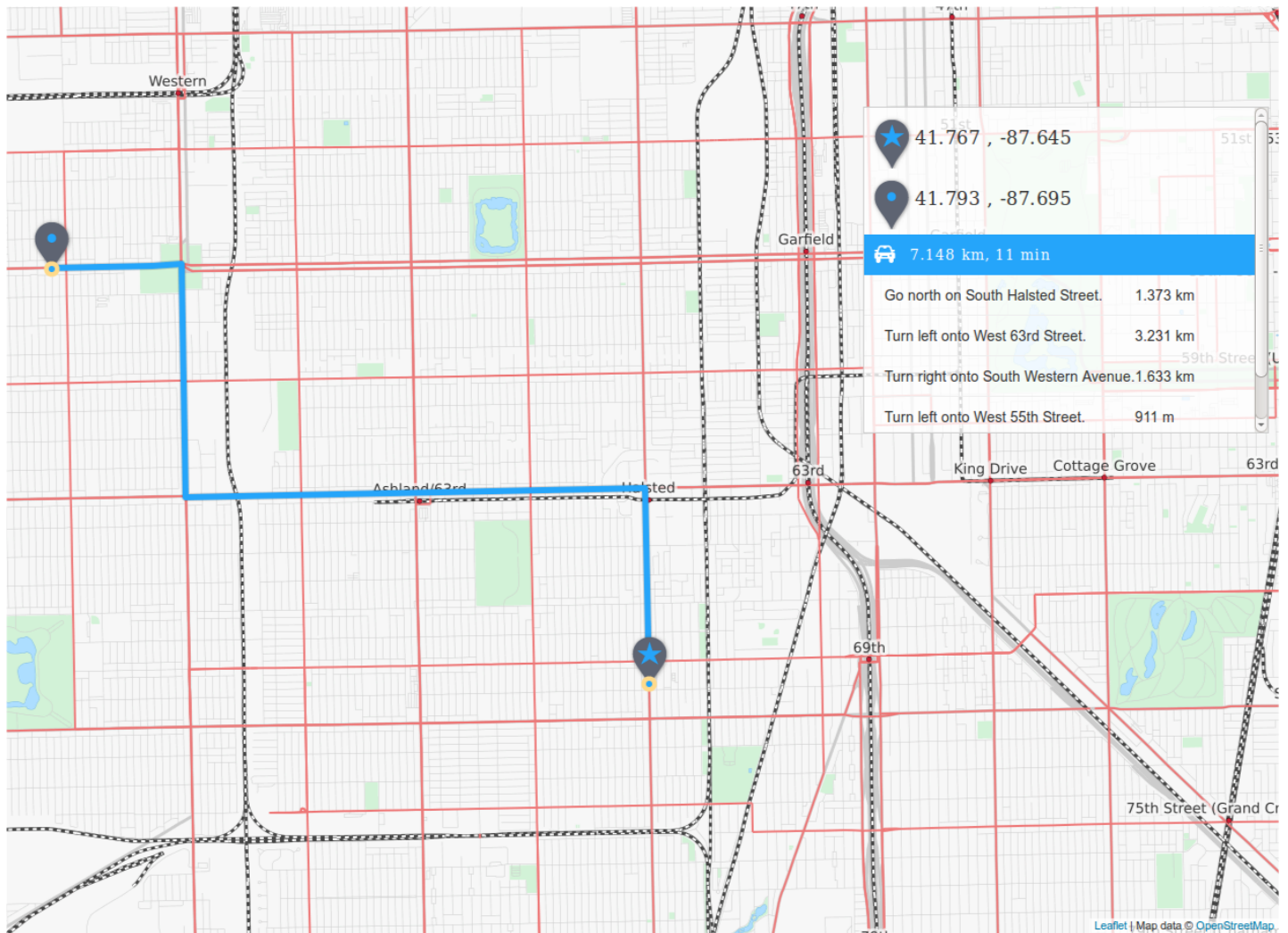
- **Turn type** - whether the turn is a left turn, right turn, or is crossing another road. The cost applied to the turn type also needs to know if driving is done on the left side or right side of the road. While left turns are generally more costly in the US than right turns, the opposite holds in UK.
- **Likelihood of stopping** - higher costs should be applied where there is a high likelihood of stopping when going from one road to another. Examples are when crossing a higher class road while on a lower class road. The opposite occurs when on a higher class road - transitions at intersections with lower class roads usually do not require a stop.
- **Name consistency** - this one is less intuitive. By applying small cost penalties when going from a road with one name onto one with a different name can lead to “simpler” route paths where there are less maneuvers or turns that need description.

In addition to these general cases, edge transition costing can be used to apply penalties for specific cases like crossing a country border, going onto a toll road, entering a road that has private access, and other cases where the route path might want to avoid specific roads or types of conditions!

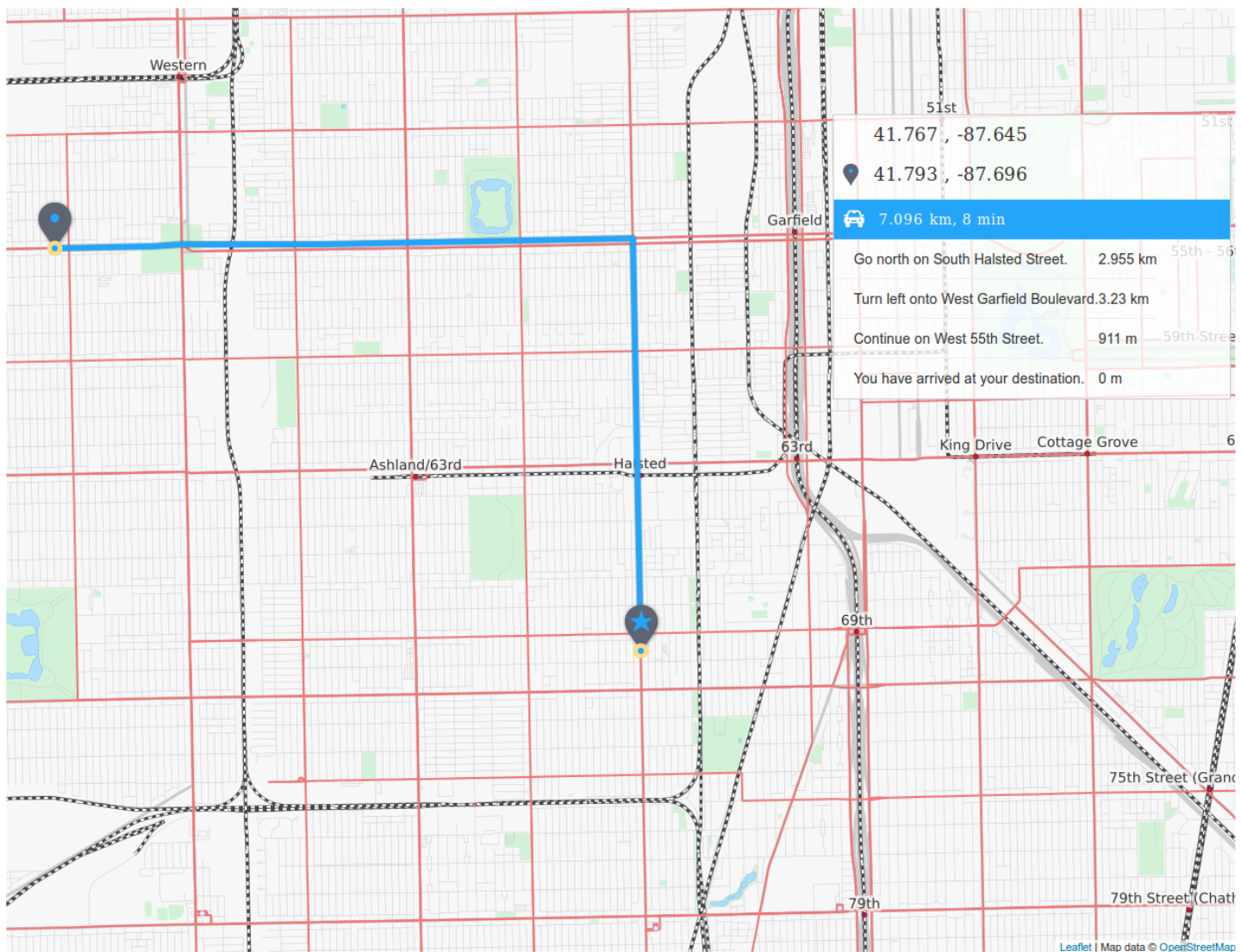
One practical example is SIF costing for driving routes (auto) which penalizes left hand turns more than right hand turns by default. (In countries where driving occurs on the left hand side of the road the opposite occurs.) The simple example below shows a case where a longer path is taken so that costly left turns are avoided. In this case, two right hand turns are made and while the route is over twice as long, they are roughly equivalent in time.



A good example of how edge transition costs can affect a route path is a driving route in a gridded downtown area. A shortest time or distance path can make many different turns as it zig-zags from the origin to the destination. Applying edge transition costs will reduce the number of turns and create a much simpler route that is often shorter time in practice. The map below is without edge transition costs - it makes additional turns that complicate the path description.



The second map shows the path with edge transition costs applied - there are fewer turns leading to a simpler route.



Note also in the first image that the estimated time is much less - this shows how important edge transition costs are to estimating the time along a route.

Check it out!

You can try out our open source routing via the **Mapzen instance of Valhalla** (<https://mapzen.com/projects/valhalla>). You can also **browse the code** (<https://github.com/valhalla>) and reach out if you have questions or suggestions!

· 04 September 2015 ·



David Nesbitt

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.

© 2017 Mapzen

Moving on up! (using Mapzen's Elevation Service)

demo (/tag/demo)

Everybody can use elevation data, and Mapzen is here to help.



How often have you ended up on a spontaneous hiking adventure on a path you've never been on before? You only live once right? You may never have the chance to take this hike again. Did it go a little something like this maybe?

- *"This won't take long at all!"*
- *"OK this is steeper than I thought but we're still making good time."*
- *"Wow this is steep."*
- (You get to the crest of a hill... only to see a valley and then another hill.)
- *"Hey, this is just how R2D2 and C3PO felt like as they got to the top of that dune and saw another dune!"*
- (No one laughs.)
- (It starts to get dark.)
- (Your significant other is no longer talking to you.)

It would have been nice to be able to see what the journey ahead was before you started. Maybe even an elevation chart to look at before deciding to plunge into your adventure? We now have some tools to help you with that on your next outing, so start clicking!

What's that journey going to look like?

Just zoom and click to add points on the map to generate an elevation chart. (Hit 'clear' to start a new route.)



Leaflet (<http://leafletjs.com>) | Maps © Thunderforest (<https://www.thunderforest.com>) | Data © OpenStreetMap contributors (<http://openstreetmap.org/copyright>)

Sampling Distance: 180m



CLEAR

permalink ([https://valhalla.github.io/demos/elevation/index.html#loc=12,47.2204,9.3883&shape=\[{"lat":47.241138,"lon":9.270758},{"lat":47.212150,"lon":9.50796}\]&resample_distance=100](https://valhalla.github.io/demos/elevation/index.html#loc=12,47.2204,9.3883&shape=[{))

The sampling distance controls how often other points than the ones you've clicked will be sampled. Clicking on a displayed marker will show its height and distance along the course of input points.

So how did we do that?

There's an API for that

To help ease your pain, we've released an **elevation service** (<https://mapzen.com/documentation/elevation/>)! The service currently responds with the height and distance along the shape used to query the elevation data source. There's also an

option to interpolate points at a regular interval along the input shape. Given that distance and height, one can visualize an elevation profile along a particular path, as shown in the widget above. All you need to get started is a series of lat,lons or an encoded polyline (from a **Valhalla route** (<https://mapzen.com/projects/valhalla>) perhaps), and an **API key** (<https://mapzen.com/developers/>).

How does it work?

We've aggregated **SRTM** (<http://www2.jpl.nasa.gov/srtm/>), **GMTED** (http://topotools.cr.usgs.gov/gmted_viewer/), and **NOAA ETOPO1** (<https://www.ngdc.noaa.gov/mgg/global/global.html>) data to provide near-global elevation and bathymetric coverage for you, meaning you can focus on integrating the data in your maps rather than configuring GDAL. Let us know where we can improve the data!

Note that the elevation service can return results not just across a valley, but also **a city** (https://valhalla.github.io/demos/elevation/index.html?#loc=13,37.747508,-122.436705&shape=%5B%7B%22lat%22:%2237.815073%22,%22lon%22:%22-122.516955%22%7D,%7B%22lat%22:%2237.697676%22,%22lon%22:%22-122.377909%22%7D%5D&resample_distance=170) or **a continent** (https://valhalla.github.io/demos/elevation/index.html#loc=5,40.880295,-96.965332&shape=%5B%7B%22lat%22:%2237.439974%22,%22lon%22:%22-132.231074%22%7D,%7B%22lat%22:%2237.683820%22,%22lon%22:%22-66.576777%22%7D%5D&resample_distance=56660)!

Check out the **elevation service documentation** (<https://mapzen.com/documentation/elevation/>) for sample requests and API documentation, or drop us a line at [routing@mapzen.com (<mailto:routing@mapzen.com>)] if you have any questions or suggestions or improvements!

*mountain goat via **rotatebox***

([https://commons.wikimedia.org/wiki/Category:Capra_ibex#/media/File:Bouquetin_des_Alpes_\(Capra_ibex\)_07.JPG](https://commons.wikimedia.org/wiki/Category:Capra_ibex#/media/File:Bouquetin_des_Alpes_(Capra_ibex)_07.JPG))

Note: This post was updated on May 22, 2017 to reflect updated data sources.

· 18 September 2015 ·

Kevin Kreiser



Kevin works on routing at Mapzen but secretly tries to work in all mapping disciplines. Er iss aa Pennsilfaanisch Deitscher.



Kristen DiLuca

Kristen is a software engineer specializing in our routing API services. She also enjoys dabbling in javascript for our open source routing test tools and always welcomes new challenges.

© 2017 Mapzen

Tilting and Tiling Ikeda

tangram (/tag/tangram) **demo** (/tag/demo)

Tangram (<https://mapzen.com/projects/tangram>) is a map made of math. But sometimes a map needs a little chaos and randomness to make it seem human.

Mapzen's Patricio Gonzalez Vivo was inspired by **Ryoji Ikeda**

(<http://www.ryojiikeda.com/biography/>) to make a map using pseudo-random functions.

Ikeda is an electronic composer and artist who “elaborately orchestrates sound, visuals, materials, physical phenomena and mathematical notions into immersive live performances and installations.”

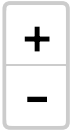


Ryoji Ikeda :: data.path, 26 SEP 2013 – 5 JAN 2014, Espacio Fundación Telefónica, Madrid, ES (<https://vimeo.com/76813693>) from **ryoji ikeda studio** (<https://vimeo.com/user10759430>) on **Vimeo** (<https://vimeo.com>).

I asked Patricio about Ikeda's work:

"It is amazing and inspiring in itself. I enjoy the process of learning from it. He use real data, big data, and remixes it and blends it in this beautifully minimalistic way that speaks so much about our digital era... one where devices hold more information that we can process, and is too easy to get lost in it."

Patricio applied three pseudo-random procedural patterns atop OpenStreetMap data to create this Ikeda-style map:



Using random can be hard; it is both too chaotic and sometimes not random enough. – Patricio Gonzales Vivo

The math behind these procedural patterns is described in chapter 10 of Patricio's "**The Book of Shaders** (<http://patriciogonzalezvivo.com/2015/thebookofshaders//10/>)", *Generative designs*.

The patterns on the streets and walls of buildings are defined using these pseudo-random functions:

<http://thebookofshaders.com/edit.html#10/ikeda-00.frag>
(<http://thebookofshaders.com/edit.html#10/ikeda-00.frag>)

<http://thebookofshaders.com/edit.html#10/ikeda-03.frag>
(<http://thebookofshaders.com/edit.html#10/ikeda-03.frag>)

<http://thebookofshaders.com/edit.html#10/ikeda-04.frag>
(<http://thebookofshaders.com/edit.html#10/ikeda-04.frag>)

Zooming out (<http://patriciogonzalezvivo.github.io/tangram-sandbox/tangram.html?styles/tilt-ikeda#11.575/37.7750/-122.4190>), you can better see the **grid** (<http://thebookofshaders.com/edit.html#10/ikeda-numered-grid.frag>) and the **number counters** (<http://thebookofshaders.com/edit.html#10/ikeda-digits.frag>) (coordinates plus seconds).

As if this were not impressive enough, Patricio added functions to simulate tilting and rotating a Tangram map within Leaflet. Leaflet is a 2D map engine, and Tangram is built on top of it, like a plug-in. We use a camera that uses matrices to skew the geometry and give the illusion of perspective. Although we don't officially support tilting in the Javascript version of Tangram like we do in **Tangram ES** (<https://github.com/tangrams/tangram-es>), it is possible to tilt the entire world's coordinates by rotating each vertex of the geometry using matrices calculations.

```
mat3 rotateX3D(float phi){
    return mat3(
        vec3(1.,0.,0.),
        vec3(0.,cos(phi),-sin(phi)),
        vec3(0.,sin(phi),cos(phi)));
}
mat3 rotateZ3D(float psi){
    return mat3(
        vec3(cos(psi),-sin(psi),0.),
        vec3(sin(psi),cos(psi),0.),
        vec3(0.,0.,1.));
}
```

We hope you are as excited about Tangram and Patricio's styles as we are. **FastCompany** (<http://www.fastcodesign.com/3051169/this-brooding-cityscape-is-what-google-maps-dreamt-about-last-night>) described Patricio's map as *"This Brooding Cityscape Is What Google Maps Dreamt About Last Night"* and **CityLab** (http://www.citylab.com/design/2015/09/a-mesmerizing-3-d-vision-of-new-york-as-pure-data/406870/?utm_source=SFTwitter) said *"Patricio Gonzalez Vivo, a graphic engineer at Mapzen, has channeled Ikeda to create a spinning, 3-D view of the city torn from a universe of pure data."*

Follow @mapzen (<http://twitter.com/mapzen>) on Twitter to stay up to date as we push the boundaries of maps with **Tangram** (<https://mapzen.com/projects/tangram>)!

· 23 September 2015 ·



Patricio Gonzalez Vivo

Patricio is an artist and graphic engineer who speaks the language of light.



John Oram

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

Add Mapzen Turn-by-Turn routing to your map

routing (/tag/routing) tutorial (/tag/tutorial)

Back in 1983 when Clark Griswold was planning his family's cross-country roadtrip as depicted in the film, *National Lampoon's Vacation* (<https://youtu.be/k5szQT71xTs>), he was using what we would consider now to be **relatively rudimentary navigation software** (<https://youtu.be/C3mvQ8V6evY>).

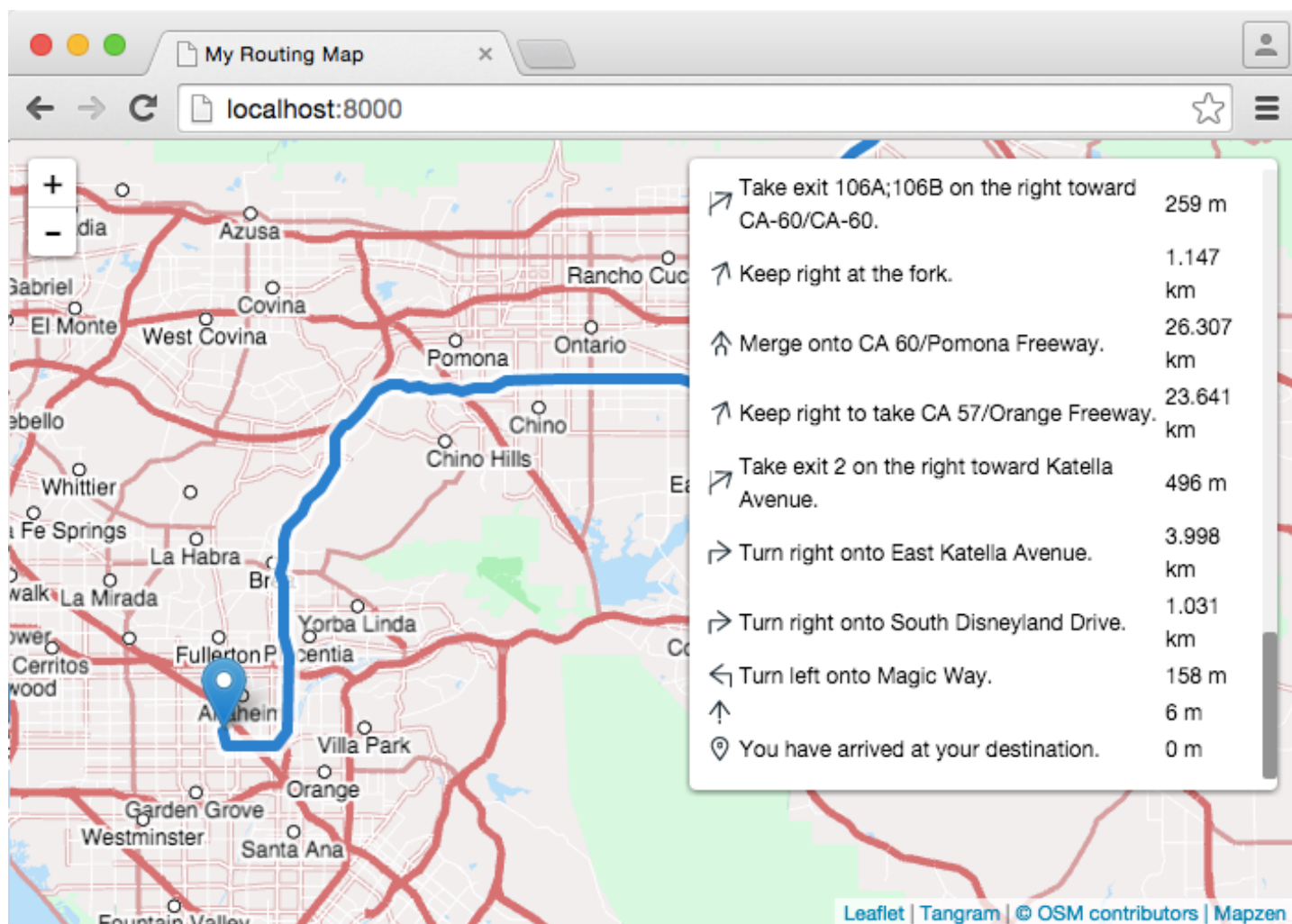


(<https://www.youtube.com/watch?v=C3mvQ8V6evY&feature=youtu.be>)

Planning the route for the Griswold family's roadtrip. Image © Warner Bros.

These days, digital maps are much more advanced. **Mapzen Turn-by-Turn** (<https://mapzen.com/projects/valhalla>), which is a routing service powered by the **Valhalla engine** (<https://github.com/valhalla>), adds routing and navigation to web or mobile applications. Mapzen's routing is free and open source, works globally, and provides dynamic and customizable routes for driving, walking, or bicycling, with clear directions for maneuvers along the route.

Mapzen has a new **tutorial** (<http://git.io/vnQ3h>) to teach you how to integrate turn-by-turn navigation with an interactive map by using routing plug-ins for the **Leaflet JavaScript mapping library** (<http://leafletjs.com/>). In the walkthrough, like the Griswolds, you will be building a map for travel by car from Chicago, Illinois, to visit a popular theme park in Anaheim, California.



As you journey through the tutorial, you will build a map that has:

- distances along your route and estimated travel times
- a route line between map locations (also known as waypoints)
- a text narrative of maneuvers to perform on the route
- functionality to drag the route start and endpoints to get a different path
- the ability change the mode of transportation, such as by car or walking

To complete the tutorial, you should have some basic familiarity with HTML and JavaScript, although all the source code is provided for you. To use the routing service, you must first obtain a free developer API key from Mapzen. Sign in at <https://mapzen.com/developers> (<https://mapzen.com/developers>) to create and manage your API keys.

You can find the tutorial at <http://git.io/vnQ3h> (<http://git.io/vnQ3h>).

· 24 September 2015 ·



Rhonda Glennon

Rhonda is Mapzen's technical publications manager and writes about maps and developer tools.

© 2017 Mapzen

The World is Yours – announcing Mapzen Search

search (/tag/search)

We live in a pretty complex world, and sometimes we need a little help to understand where we are. We're excited to announce **Mapzen Search**, our new search engine for places that takes our philosophy of open communities creating data and code to its heart. Mapzen Search will launch in the next week, and in the meantime, we wanted to share some background on why the time is right for a new, open search engine for places.

*Note: If you've been a user of the Pelias web service during our beta, you'll have to make a few changes as we transition from Pelias beta to 1.0 of Mapzen Search. You may want to **skip to this section** to understand the changes which we hope are relatively straightforward.*

We live in two worlds at once. There's a human world where we call places by their addresses and names. Then there's another world for computers in which every place is represented through geographic coordinate systems like latitudes & longitudes or **geohashes** (<http://www.bigfastblog.com/geohash-intro>). Dozens of times every day we cross that boundary between the world of names and the world of coordinates and it's all facilitated by a process called *geocoding*. In the coordinate world, it's easy to do things like find things nearby, measure distances and correlate data, but to get there takes a leap through a boundary – and every time that boundary is crossed there's a little toll paid. Do it enough and it adds up. But it also comes with all manner of restrictions of how you can see the world.

But what if it didn't have to be this way? Mapzen Search is our answer to this call. It's powered completely by open data, available freely to everyone, and importantly open to all to make better. It's also powered by Mapzen's open source geocoding project, Pelias. We're building it in the open, and is **open to collaboration from all** (<https://github.com/pelias/pelias>) to make searching our world better and smarter.

Mapzen Search is powered completely by open data, available freely to everyone, and importantly open to all to make better.

We're still grappling with the implications of these tools, but searching for places has yet to be truly democratized. To search the globe at a scale where you can afford to take things for granted is expensive. It also generally means searching through the lens of a corporation's **owned copy of the world** (<http://www.newrepublic.com/article/122083/pax-google-how-your-friendly-search-engine-took-over-world>) where the data powering it belongs to them, as do the tools that make the data useful.



(<http://digitalcollections.nypl.org/items/510d47e2-0b65-a3d9-e040-e00a18064a99>)

The Time is Now

We have seen efforts to correct this in the past, with OpenStreetMap's community driven **Nominatim** (<http://nominatim.openstreetmap.org>) project. It's a remarkable open geocoder, but is challenging to run at scale without access to some impressive hardware, which makes customizing it extremely difficult (*and is also part of the reason Nominatim is unable to offer autocomplete services*). The open community genuinely loves Nominatim and, unfortunately, last

month the largest provider of a free Nominatim service, Mapquest Open, announced it would **start charging high fees** (<http://devblog.mapquest.com/2015/08/17/mapquest-free-open-license-updates-and-changes/>) for even moderate use of the service.

We believe that the time is right for a new, open experimental geocoder to give everyone a world-class starting point to search the world.



(<https://collection.cooperhewitt.org/objects/18635643/>)

Here's How

The world is a complex place. Rules about how places are organized are only relevant for certain areas, under certain conditions, and are full of exceptions.

Much of the hard work is in collecting the data so it can represent the world correctly which, thanks to **OpenStreetMap** (<http://openstreetmap.org>), **OpenAddresses** (<http://openaddresses.io>), and other open data efforts, gives us the greatest accumulation of openly licensed place data the world has ever known. These are projects that give their data to all comers and in the case of OpenStreetMap enable anyone to correct and improve the underlying data, making the tools that use it, and thereby the world just a little bit smarter every time.

So the data collection effort is well underway, and search is a synthesis of this data and how we interpret it. Search requires knowing things like:

- UK postcodes beginning with *E* represent East London, same for *S*, *N* and *W*.
- Different countries have different address numbering schemes; some 'zig-zag' up the street while others go up one side and down the other.
- 0 is a valid street number in some countries.
- US highways with even numbers run *East-West* and odd routes run *North-South*.
- Most countries have a clear hierarchy of country/city/state etc. but there are some odd exceptions such as Vatican City.

We believe that an open geocoder can only grow over the long-term if it's easy to learn as a community and incorporate truly representative local knowledge.

To account for that, you can either hire an army of employees to conduct the research in every far-flung corner of the world, or rely on local knowledge of users around the world to help create open services that are genuinely representative. Mapzen is committed to working with folks in the open mapping community and beyond to make the Pelias engine incrementally better at representing the world the way it really is, through the input of the very people that occupy it.

As we start, our focus is making Mapzen Search & Pelias a project where participating in the design and development is fundamentally open. It's why we're building Pelias with Node.js and Elasticsearch.

With Node.js, we're able to open up Pelias' development to one of the largest development communities in the world. Javascript has become a *lingua franca* of programming that has not only shown itself as powerful for building tools, but for building tools that garner powerful communities. Node.js has a huge collection of modules which are shared with other users in the community through the package manager *npm*. As we develop functionality for Pelias and Mapzen Search, we publish each piece of discrete functionality to npm so that the community can also benefit from it, and contribute features back to us.

And while we orchestrate in Javascript, we power search with *Elasticsearch*, another remarkable open source project that lets us scale Pelias' search capabilities from running on just a single desktop to across a data center. It's a framework that'll let us continue to build features present in Pelias and Mapzen Search that simply aren't there in other open geocoders, like autocomplete. It lets us to easily adapt the strategies we use to search for things and what we can offer with those searches.

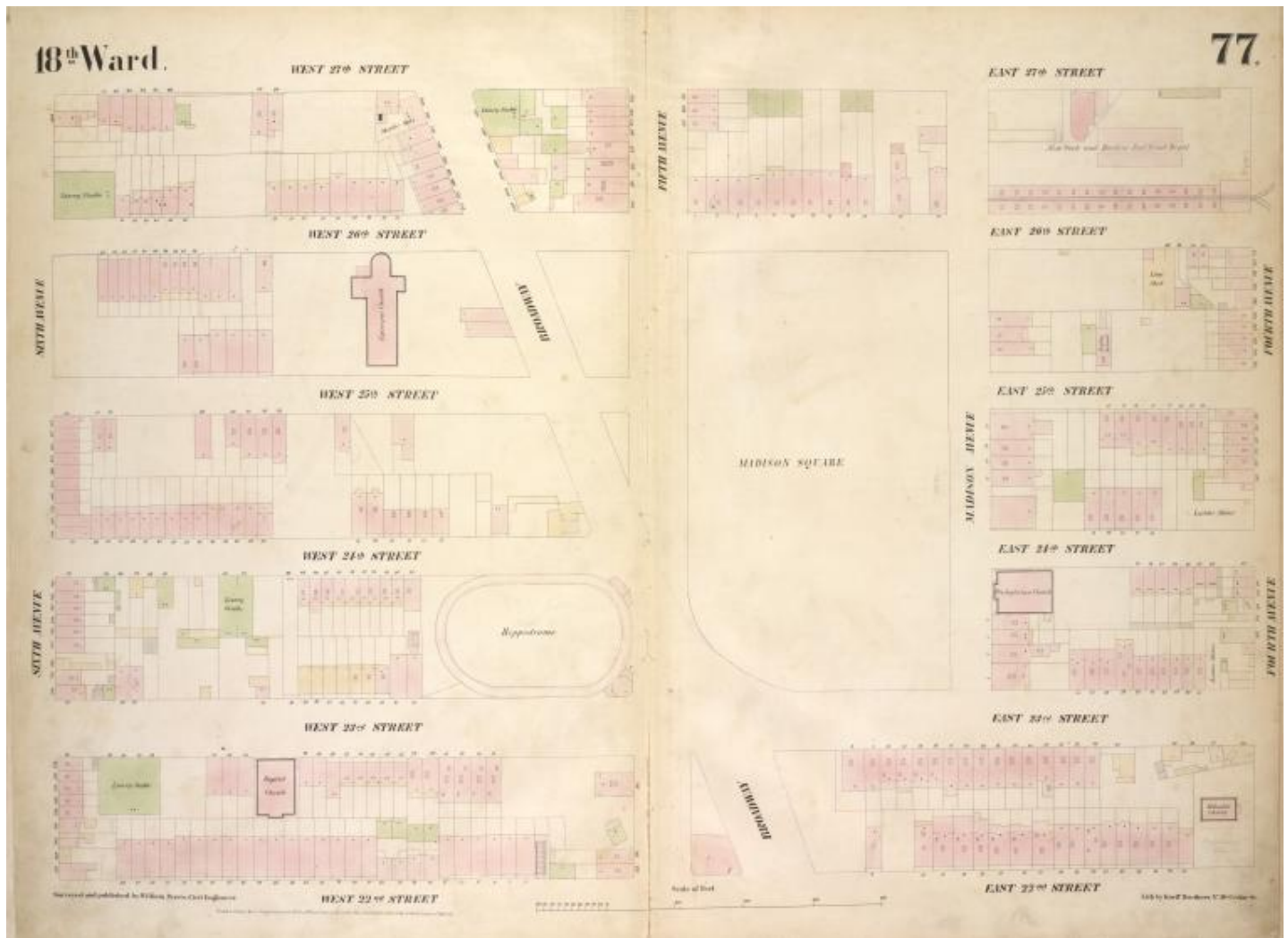
We think open data, open source, and open strategy win over proprietary solutions at any part of the stack and want to ensure the services we offer are in line with that vision.

We want to give everyone a world-class starting point to search the world. With the revolution in open data being led by communities, and the Open Government movement, anyone who can run Pelias can have a powerful geocoder out of the box based on open data. That's why we're building Mapzen Search, our flagship use of Pelias to run entirely on open data - we want to ensure the baseline you get with us is as good as it possibly can be. We think open data, open source, and open strategy win over proprietary solutions at *every* part of the stack and want to ensure the services we offer are in line with that vision.

You've got this data, shouldn't you be able to use it on any map you want?

At the same time, we want to open the domain of place search and geocoding beyond where it is today, and give others a powerful place to start in customizing a geocoder for specific domains, companies, and purposes we haven't even imagined today.

We couldn't be more excited for you to start searching the world with us.



(<http://digitalcollections.nypl.org/items/510d47e4-56ba-a3d9-e040-e00a18064a99>)

Upcoming Changes

The release of Mapzen Search is a major milestone that we hope will make Pelias easy for developers to start using immediately and to support the improvements we have in store over the next few years. However, it also introduces changes which are incompatible with the beta some developers have been using. We will continue to support calls to the existing API for at least the next two months (through November) to give developers time to transition to Mapzen Search. We realize that any partner API change, no matter the size, is a burden on existing users, and so making this change is something we do not take lightly.

The timing of this transition only immediately affects current users of **pelias.mapzen.com** (<https://pelias.mapzen.com>). If you are running your own instance of Pelias, you can control timing of the transition by choosing when to roll out the upgrade.

We are formalizing our documentation for this new API. But if you're an existing Pelias user, we should talk. Because we haven't previously required registration for Pelias, we can't get in touch with folks directly.

If you use Pelias today, please let us know by filling out **This Form**

(https://docs.google.com/forms/d/1X9zG9P_WMjgUGZ9-afi34iY-Vpqt_34nb2f8c-5OdMA/viewform?usp=send_form), or sending us an email at **search@mapzen.com**

(<mailto:search@mapzen.com>). We want to get in touch to make this transition as easy as

possible for you and your users.

More soon!

Image credits:

- Lionel Pincus and Princess Firyal Map Division, The New York Public Library. "Orbis terrae compendiosa descriptio : quam ex magna universali Gerardi Mercatoris Domino Richardo Gartho, geographie ac ceterarum bonarum artium amatori ac fautori summo, in veteris amicitie ac familiaritatis memoriam Rumoldus Mercator fieri curabat A0. M.D. LXXXVII." *The New York Public Library Digital Collections*. 1637.
<http://digitalcollections.nypl.org/items/510d47e2-0b65-a3d9-e040-e00a18064a99>
(<http://digitalcollections.nypl.org/items/510d47e2-0b65-a3d9-e040-e00a18064a99>)
- Textiles department of Cooper Hewitt, Smithsonian Design Museum. "Jacket, Wearin' The States, 1990. tyvek, a dupont copyrighted fabric, other synthetics. Gift of Interarts. 1991-14-6." *Cooper Hewitt Collections*. 1990.
<https://collection.cooperhewitt.org/objects/18635643/>
(<https://collection.cooperhewitt.org/objects/18635643/>)
- Lionel Pincus and Princess Firyal Map Division, The New York Public Library. "[Plate 77: Map bounded by West 27th Street, East 27th Street, Fourth Avenue, East 22nd Street, West 22nd Street, Sixth Avenue.]" *The New York Public Library Digital Collections*. 1852 - 1854.
<http://digitalcollections.nypl.org/items/510d47e4-56ba-a3d9-e040-e00a18064a99>
(<http://digitalcollections.nypl.org/items/510d47e4-56ba-a3d9-e040-e00a18064a99>)

Special thanks to Stephen Hess, Peter Johnson, Nathaniel Vaughn Kelso, Randy Meech, John Oram, Julian Simoni, Kaitlin Thaney, and Diana Shkolnikov for reviewing drafts and generally helping make this piece better.



David Riordan

Former product manager for Mapzen Search. Historical mapping, open tools, open access, and open data.

© 2017 Mapzen

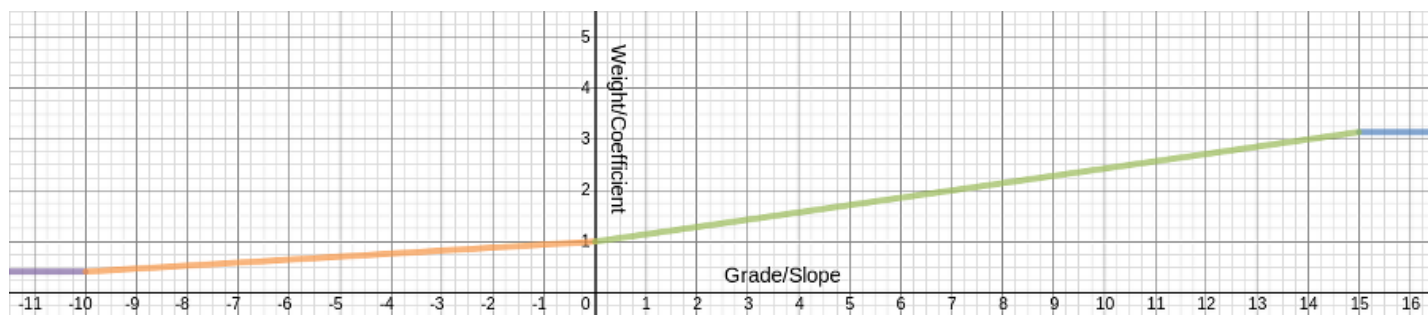
Making the Grade — Worldwide Elevation-Influenced Bicycle Routing

Mapzen and the Valhalla team recently announced updates to their bicycle routing service: **Bicycle Routing with Valhalla** (<https://mapzen.com/blog/valhalla-bicycle-routing-options>). Now we have taken it to greater heights by adding “elevation-influenced” bicycle routing. We are pleased to announce the addition of elevation and grade factors into the bicycle costing model worldwide. How badly do you want to avoid those steep hills on your next ride?!

Bicycle routing presents several unique challenges. Among the challenges are the wide range of user abilities, preferences, and equipment. Bicyclists vary in their experience and comfort level using roadways shared with larger vehicles (e.g. automobiles, buses, trucks). Different bicycle types are more or less suitable to the wide variety of road and path surfaces. The wide range of physical ability of bicyclists also comes into play when selecting routes in locations where hills and steep grades might be present. The Valhalla team at Mapzen feels these factors all play into one of the strengths of the Valhalla routing engine - its dynamic, run-time costing module known as **Sif** (<https://github.com/valhalla/sif>). Further background on Sif can be found in our prior blog post: **Dynamic Costing in Valhalla** (<https://mapzen.com/blog/dynamic-costing-via-sif>).

Addition of Elevation Factors into Valhalla Routing Tiles

In order to measure the change in elevation over a given segment of road, we’ve built a library (and service) called **Skadi** (<https://github.com/valhalla/skadi>); the goddess of the mountains. Skadi has the ability to efficiently query worldwide digital elevation model data. We use this library when building routing tiles to estimate the prevailing grade/slope of a given section of road. We call this the weighted grade. Here’s how it works.



Given a segment of road, we evenly sample points (at 60m apart) along it. At each sample, we measure the elevation. We then compute the grade/slope between each pair of sample points and weight it using the above function. This is essentially a linear combination designed to approximate the overall grade/slope of a given segment of road. You'll notice that sections with higher upward slope are weighted more and that conversely higher downward slopes are weighted less.

The intuition is that steeper sections will require more "cost" to traverse whether walking, biking or driving. If you're biking the unit of cost might be time, since you can't ride as fast up steep hills. When driving the unit of cost might be the fuel used to overcome inertia. In any case, we are attempting to measure approximately how much energy might be needed to traverse a given section of road so that we can minimize (or maximize!?) it, along with other factors, when computing your path!

Adding Grade Factors to Dynamic Costing

The weighted grade is used within bicycle costing in 2 ways:

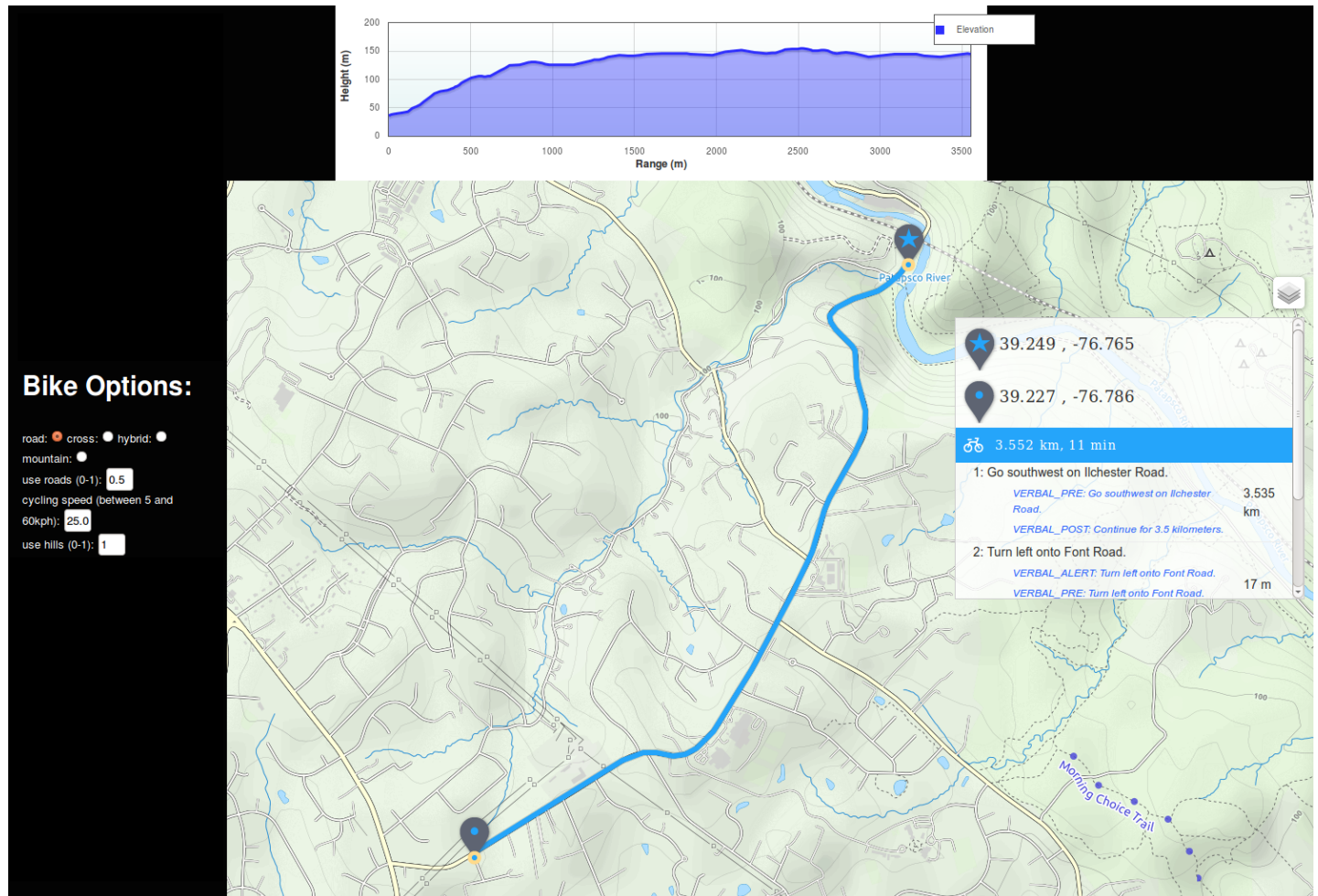
Weighted grade impacts the speed and resulting time along an edge. The default speed used for bicycle routing is assumed to be the average speed the bicyclist can maintain on level grades over the length of the route. This speed is modulated based on the weighted grade: increased for grades indicating a descent, and decreased for uphill grades. The higher the weighting factor the steeper the "average" grade along the edge, resulting in a higher reduction of speed results. Thus, the weighted grade of an edge impacts the time along the edge and helps avoid steep grades. It also helps provide a better estimate of the actual time along the route. In general, routes with hills take longer than routes on level ground.

A second use for weighted grade considers the bicyclists desire to avoid or use hills in the route. A single option called `use_hills` has been added to the Valhalla bicycle costing module. This option is similar to the `use_roads` option. It is a value from 0 to 1.0 indicating the bicyclists comfort level with hills and steep grades. A value of 1.0 indicates a strong, experienced cyclist who does not mind a path with hills. A value of 0.0 indicates the cyclist wishes to find paths that try to avoid hills. Based on this fact, extra cost (penalties) are applied to edges based on the grade. Note that penalties are applied to downhill grades as well! The old adage among cyclists is that what goes down, eventually must go up so after every long downhill there will eventually be a long uphill!

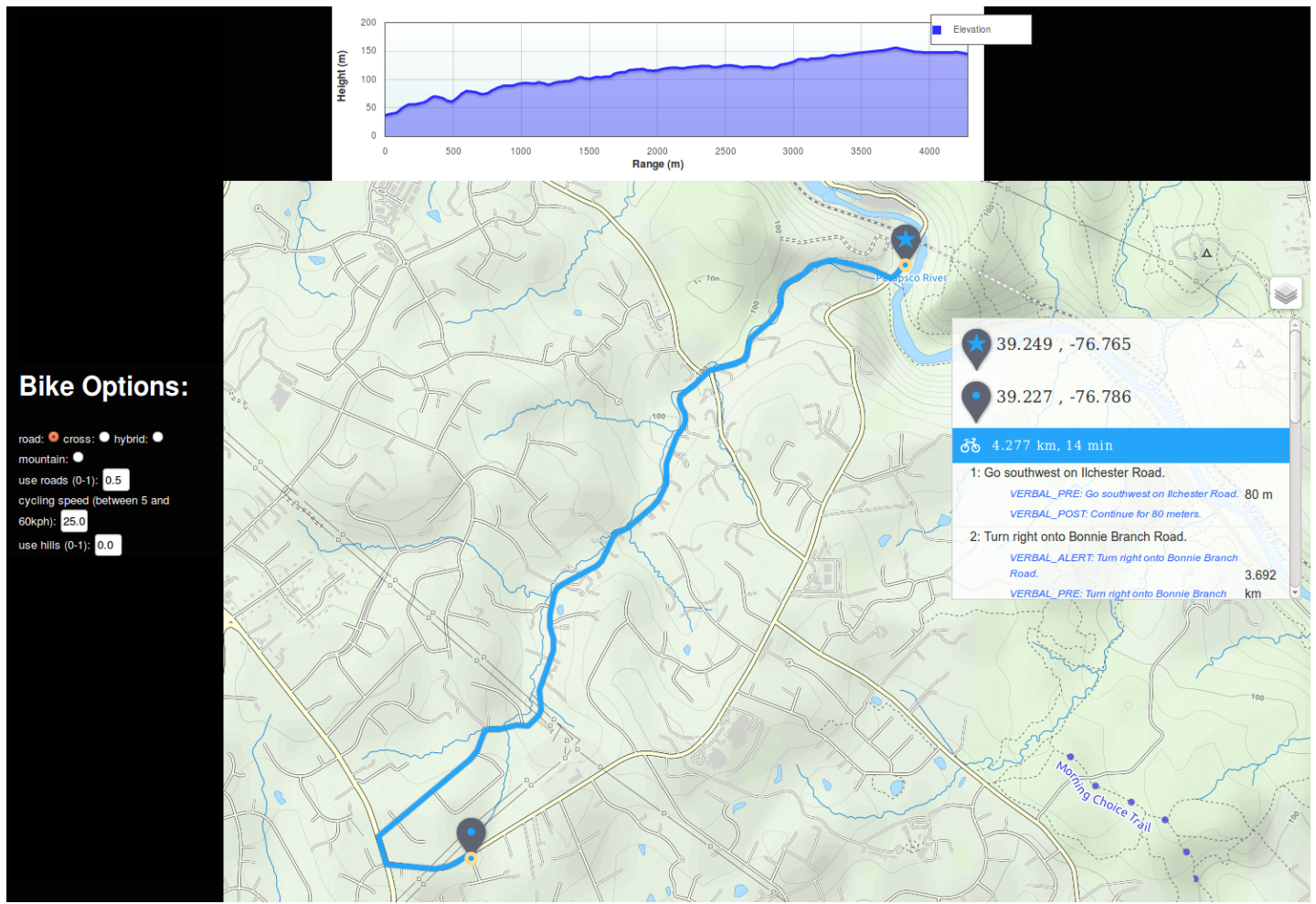
Examples

The following cases illustrate a set of hills I used to ride in my early days of cycling. This is an area near Ellicott City, MD starting at a point on Ilchester Road along the Patapsco River and ending near the end of Ilchester Road close to Montgomery Road.

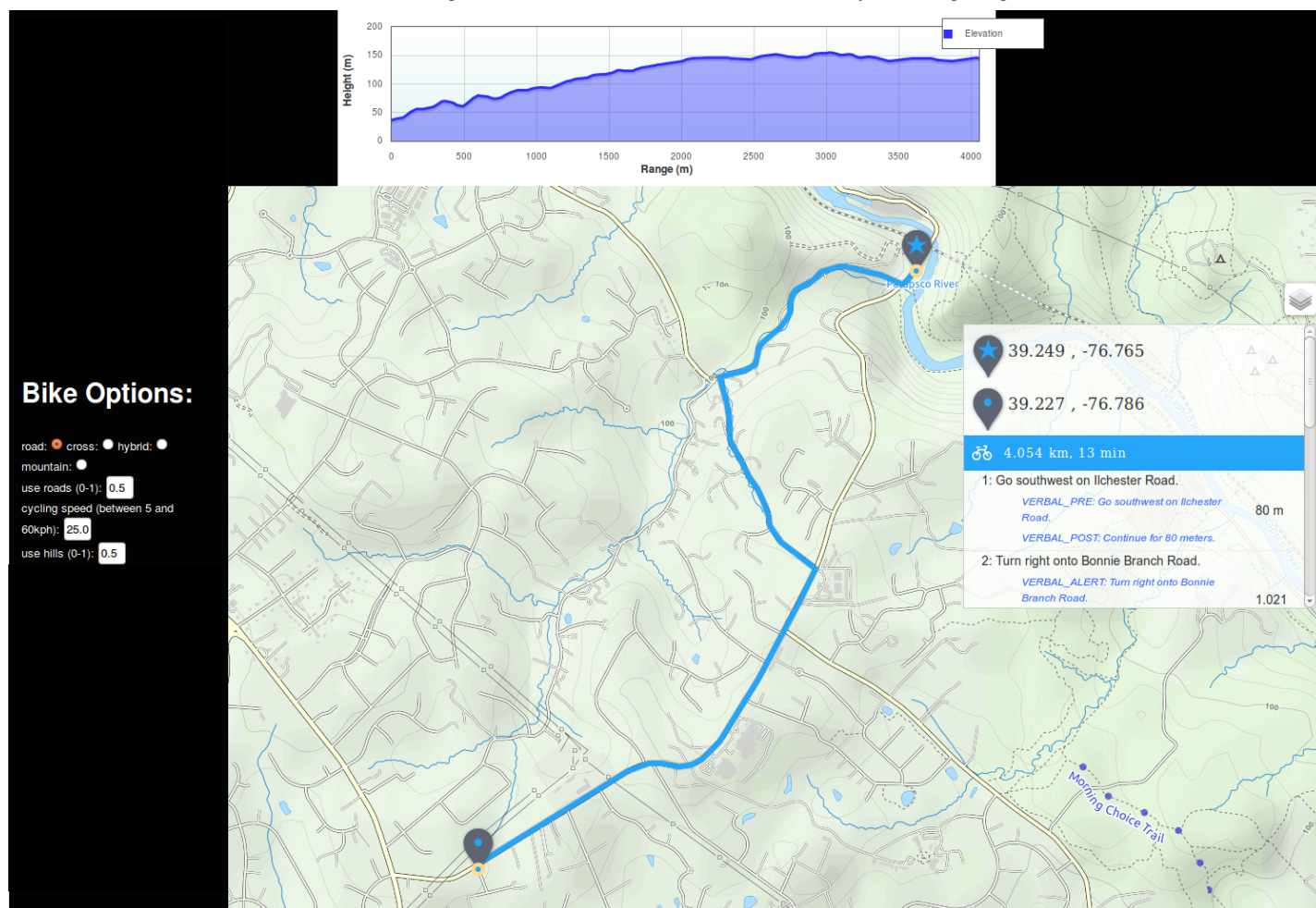
The shortest path is to head straight up Ilchester Road. The problem is the initial part of the hill is roughly 16-18% for the first 400 meters. A hill-climb time trial race is held on this hill, so it is only for strong riders or those looking for a challenge. After the initial steep section, Ilchester Road continues with a more gradual climb and then levels off. With a `use_hills` factor of 1.0 the shortest distance and shortest time path is taken - straight up the steep hill.



For those with less inclination to tackle steep grades, we can try to find an alternate path by setting the `use_hills` factor to 0.5. This might be for someone who doesn't want to go too far out of the way to avoid a hill. In this example the path avoids the steep section of Ilchester Road, instead choosing a less steep uphill path (Bonnie Branch Road onto Beechwood Road then back onto Ilchester) to get to the top of Ilchester hill. Here the elevation slowly rises from 30 meters to 150 meters over approximately 2km. This is an average grade of 6% - still uphill but much less strenuous.



For a novice bicyclist or one who wishes to avoid hills, we can try a `use_hills` factor of 0.0. Here, a longer path up the more gradual climb along Bonnie Branch Road (approximately 5% average grade) is taken the entire way to Montgomery Road and then the path backtracks to get to the destination along Ilchester Road. Here the route is nearly 1km longer than the shortest route (4.35km vs 3.5 km) and takes 3 minutes longer (14 minutes vs. 11 minutes), but with the extra cost penalties applied to the steeper grades the path ends up along the most gradual path out of the river valley.



The sample images here were created using a combination of Valhalla routing including the elevation-influenced bicycle route API as well as the new **Mapzen Elevation Service** (<https://mapzen.com/blog/moving-on-up>) API to get the range/height profile information to create the elevation chart. The output route path information (an encoded string of latitude,longitude positions along the path) is sent directly to the elevation service to get the heights along the path for charting.

Check it out!

You can try out our open source routing via the **Mapzen instance of Valhalla** (<https://mapzen.com/projects/valhalla>). You can also **browse the code** (<https://github.com/valhalla>) and reach out if you have questions or suggestions!

· 28 September 2015 ·



David Nesbitt

Dave leads Mapzen Mobility engineering. Rides a variety of 2 wheel vehicles.

© 2017 Mapzen

Spelunker – Jumping into Who's On First

data (/tag/data) whosonfirst (/tag/whosonfirst)



Who's On First

Aaron Straup Cope / FOSS4G September 2015

*The following are the presenter notes for a talk I did about Who's On First and the recently released Who's On First spelunker at the **Free and Open Source Software for Geospatial** (<http://2015.foss4g.org/>) conference, in Seoul.*

Hello. It's nice to be back at **FOSS4G** (<http://2015.foss4g.org/>). The last time I was here was in 2007, **when FOSS4G was held in Victoria** (<http://2007.foss4g.org/>). In between life and circumstance have prevented me from being able to return until now so it's a pleasure to be speaking to you today.

Editor at Large, Mapzen

@thisisaaronland

My name is Aaron and **I work at Mapzen (<http://www.mapzen.com/>)**. One of the things this talk has highlighted is that no one at Mapzen has any idea what my official title is. For now we're trying out "Editor at Large" and we'll see how that goes.

I'd like to begin by doing a quick review of the last ten years of my work-life in three short slides by way of introduction and to set the stage for the rest of this talk.



A million years ago I worked at **Flickr** (<http://www.flickr.com/>), the photo-sharing website. One of the projects I was involved with was **geotagging** (<https://blog.flickr.net/en/2006/08/28/great-shot-whered-you-take-that/>), or the ability to attach a location to your photos. Geo at Flickr encompassed many features and even more moving parts but one of the things to emerge from all that work were the so-called “**Alpha shapes**” (<https://code.flickr.net/2008/10/30/the-shape-of-alpha/>) shown above.

These were geometries for all the places in the Flickr hierarchy – countries all the way down to neighbourhoods – derived entirely from geotagged photos. As you can see some of these shapes were often weird and greedy in their perspective but rarely were they entirely wrong either.

Most importantly though **we released these shapefiles as a CC0 dataset** (<https://code.flickr.net/2011/01/08/flickr-shapefiles-public-dataset-2-0/>) because nothing of that granularity with a permissive license had existed before.



After Flickr, I worked at **Stamen** (<http://www.stamen.com>), a San Francisco design studio focussed on data visualization and maps. Lots and lots of maps. Stamen has always been about more than just maps but during the years that I was there we did a pretty good job of giving people the impression that it was mostly about maps.

Stamen came of age during a time when people were just starting to become aware of the amount of data accrued by almost everything an individual or an organization did. Almost all of Stamen's work, at some level, has been about **reveling in the availability** (<https://vimeo.com/112308535>) of all the data and celebrating the opportunities it presents.

This is one of the projects that I worked on at Stamen called **Prettymaps** (<http://prettymaps.stamen.com/>). Prettymaps began life as a way to stress-test a particular piece of software by throwing as much data as we could at it and then watching what happened.

In the end Prettymaps was created using only a small slice of all the available data – alpha shapes from Flickr, roads (and on/off ramps) from **OpenStreetMap** (<http://www.openstreetmap.org/>) and urban areas from **NaturalEarth**

(<http://www.naturearthdata.com/>) – but **this is what happened**
(<http://prettymaps.stamen.com/201008/about/>) when we put them all together.



More recently, I took a little bit of a detour into the museum world. Until June of this year I worked at the **Smithsonian Cooper Hewitt Design Museum** (<http://www.cooperhewitt.org/>) in New York City as part of **the Digital and Emerging Media team** (<http://labs.cooperhewitt.org>).

The details of the work at the Cooper Hewitt are **an entirely other story** (<https://collection.cooperhewitt.org/pen/>). The short version is that we designed and built custom electronic hardware and all the backend software and infrastructure to allow visitors to collect objects on display in the museum and keep a permanent record, on the web, of their visit.

In short, we built **the world's most complicated bookmarking system** (<http://labs.cooperhewitt.org/2015/exporting-your-visits/>).



I like to think that what connects all this work is a focus on networks of documents. A network of documents is hardly a new idea. In fact, what I've just described is the web, **circa 20 years ago** (<https://adactio.com/journal/6507>).

There are moments, these days, when it seems as though the web is being actively recast as little more than the transport mechanism for an even more expansive television culture so maybe it's worth taking a moment to remember **why the web was, and remains, important** (http://idlewords.com/talks/what_happens_next_will_amaze_you.htm).

First, the importance of recall.

Recall has always been a power dynamic (<http://www.aaronland.info/weblog/2014/09/11/brand/#dconstruct>) and the ability for people to visit, or revisit, a "thing" on the web with the luxury of choice about *when* they do that is pretty remarkable. Kind of like a library, the original network of documents, but not constrained by the details of physical space.

Second, the patience of use.

This is a short hop from the importance of recall but the burden is shifted from the “consumer” to the “producer”. The web makes it possible in very real and practical terms to put something online in a way that doesn’t require it to enjoy immediate mass-appeal in order to recoup the costs of that effort.

The costs of an effort are not always measured in financial terms which is another way of saying that people have always been willing to “throw money away” in the service of a project or an idea and the web has made this choice available to more people, in more places, more than possibly any time before it.

Finally, that the value of the aggregate is greater than the sum of a perfect subset.

This is not an absolute truth but it was the lesson we proved when we rebuilt the collections website at the museum. Museums don’t like to talk about this but, on the whole, they all have *terrible* metadata. For any given museum only a tiny fraction of their holdings have been fully vetted and properly documented. Although the Cooper Hewitt is set to have **fully digitized their collection** (<http://labs.cooperhewitt.org/2015/long-live-rss/>) by mid-2016 when I started only about *seven percent* of the collection had been photographed. This is still the norm for most museums.

Mostly, any kind of collective proof – photographs, metadata or any kind of documentation about why an object has been collected – exist only as anecdotes or institutional mythology. What the Cooper Hewitt did in 2012 was to **publish the entirety of its collection metadata, mistakes and all, as a CC0 dataset** (<https://www.github.com/cooperhewitt/collection/>) and then proceed to rebuild the collections website around it.

Rather than being a cathedral of emptiness (but perfect emptiness) the collections website is more like a big noisy cocktail party. It contains ample threads for casual users and scholars alike to find their own meaning in the collection.



Fast-forward to today and Mapzen is building a gazetteer of places, built from and published under a permissive license. It's called **Who's On First** (<http://whosonfirst.mapzen.com/>).

The easiest way to think about a gazetteer is that it is a big list of places, each with a stable identifier and some number of properties describing that location and relationships to other documents in the gazetteer.

Or: A network of documents.

There are a number of reasons for building a gazetteer. One of them is that we are erecting a scaffolding for place around the other map and geo related services we are building at Mapzen. What happens when every service returns a stable identifier for each place it references in its result set that can handed off to another service for more details about that place?

Despite what New Yorkers might tell you there are in fact **12 different towns called Brooklyn** (<http://whosonfirst.mapzen.com/spelunker/search/?q=brooklyn&placetype=locality>) out there in the world and our hope is for people to be able to use the gazetteer as a conversational

shorthand for place without always having to disambiguate *which* place is being talked about.

This is the goal and make no mistake: Our goal is to succeed. That said, we are not the first organization to build tools and services around open source and open data, though, and we may not be the last.

One of the things we actively consider at Mapzen is: *How do we ensure that our work can outlast our endeavour?*

This question influences many of the decisions – the first principles – we have made about the gazetteer.



**the data is not the
database**

I am only going to touch, briefly, on a few of the decisions we've made because it's a complex subject and this is a short talk. The first and in many ways the most important of those decisions is: The data is not the database.

Databases come and databases go. There are lots of databases out there and they are all good at different things. We are living through an interesting time where a lot of “innovation” in database technology seems to be about mostly aping the features in other databases. Often this comes at the expense of performance and where it doesn't it often pays for that performance in complexity.

The real issue though is not any one particular database but an attempt to ensure that Who's On First demands as small an infrastructure burden as possible. To that end we want to make sure that Who's On First works with all the databases (or at least most of them).



portability

So we have standardized on text files. Plain old text files because all computers can handle text files. Every place in the gazetteer is a single stand-alone text document, specifically a GeoJSON document. We chose GeoJSON because it is the least amount of markup, today, for structured geographic data and because there are lots of tools for converting GeoJSON in to all the other formats that people doing geo-work know and love.

The decision to settle on text files does make things difficult in the short-term. Storing and processing as many files as we're generating **can be problematic**

(<http://githubengineering.com/counting-objects/>) and definitely introduces its own kind of infrastructure burden. The advantage we have is that it's 2015 and we are rich in processing power so it seems like a manageable problem.

Part of the work in the short-to-medium term will be thinking about how we bundle up all the source data in to formats and packages to make it easier to distribute the data and for people to get started *doing* something with it.

stable permanent IDs

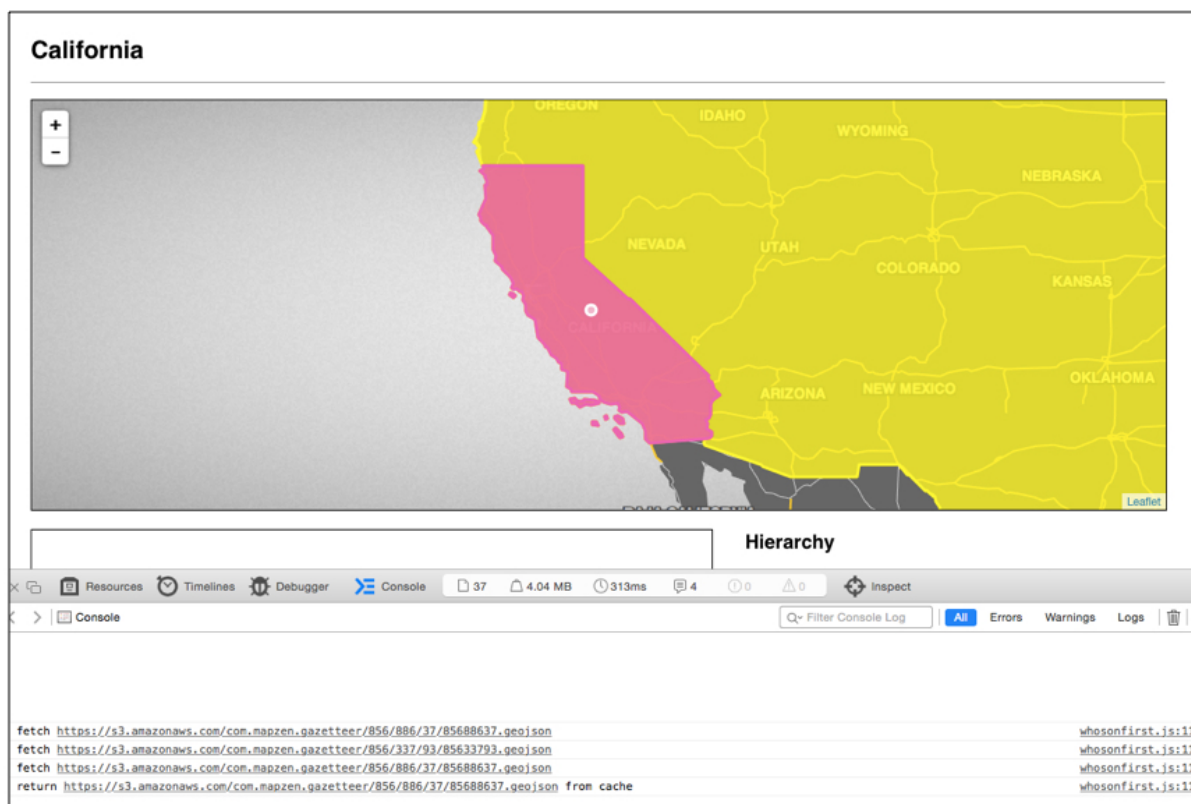
As I've alluded to every record has a stable permanent identifier and by extension URL (derived from that ID).

We have also made a point of storing all the paths to individual documents as relative paths. Currently we are the canonical source of truth for all this data hosted at

`whosonfirst.mapzen.com/data` but if you need to host your own version of the Who's On First

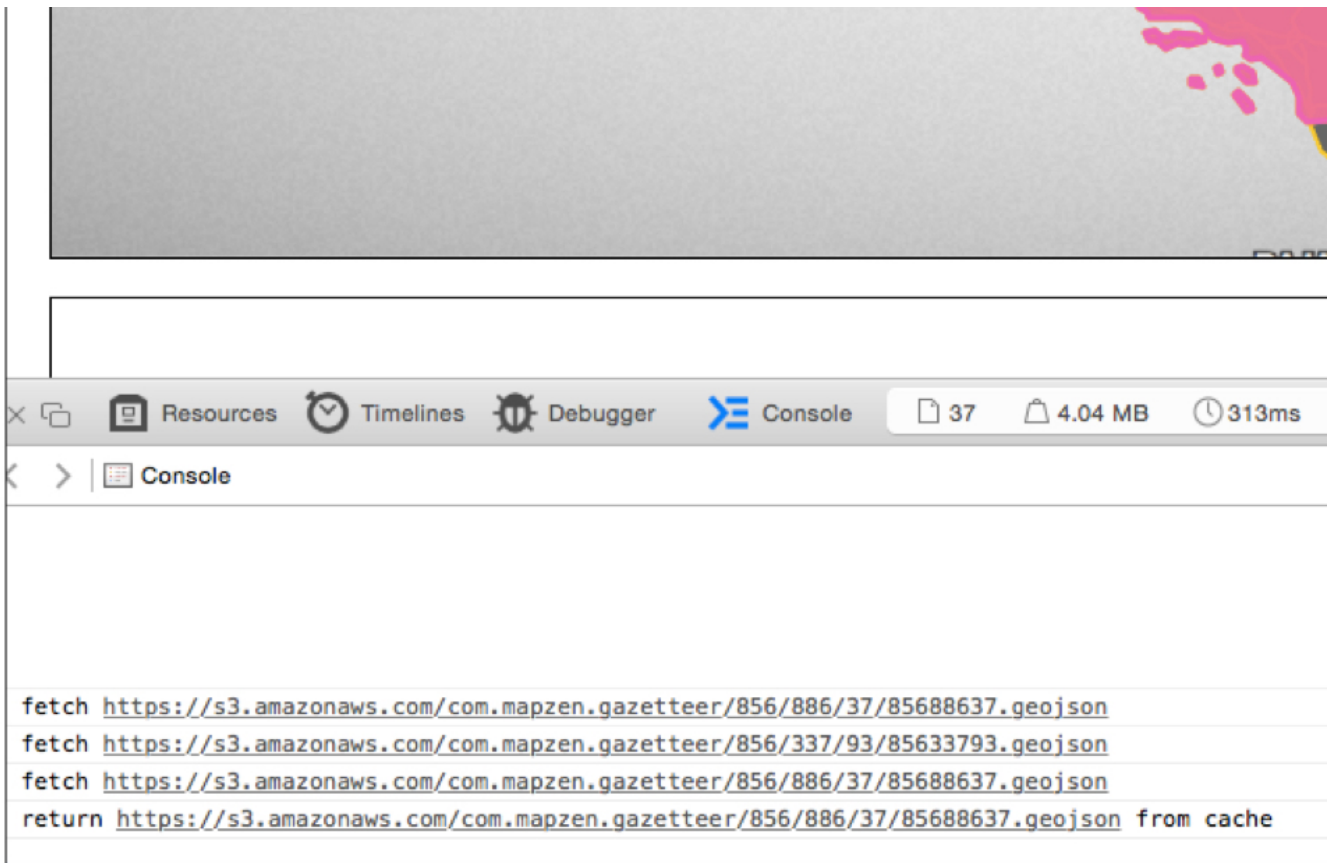
data you shouldn't have to do anything besides changing the prefix, or domain, indicating where the data lives.

One of the things that we've been actively testing is using all those IDs and URLs and pointers to *other* IDs to treat the network itself as a kind of database.



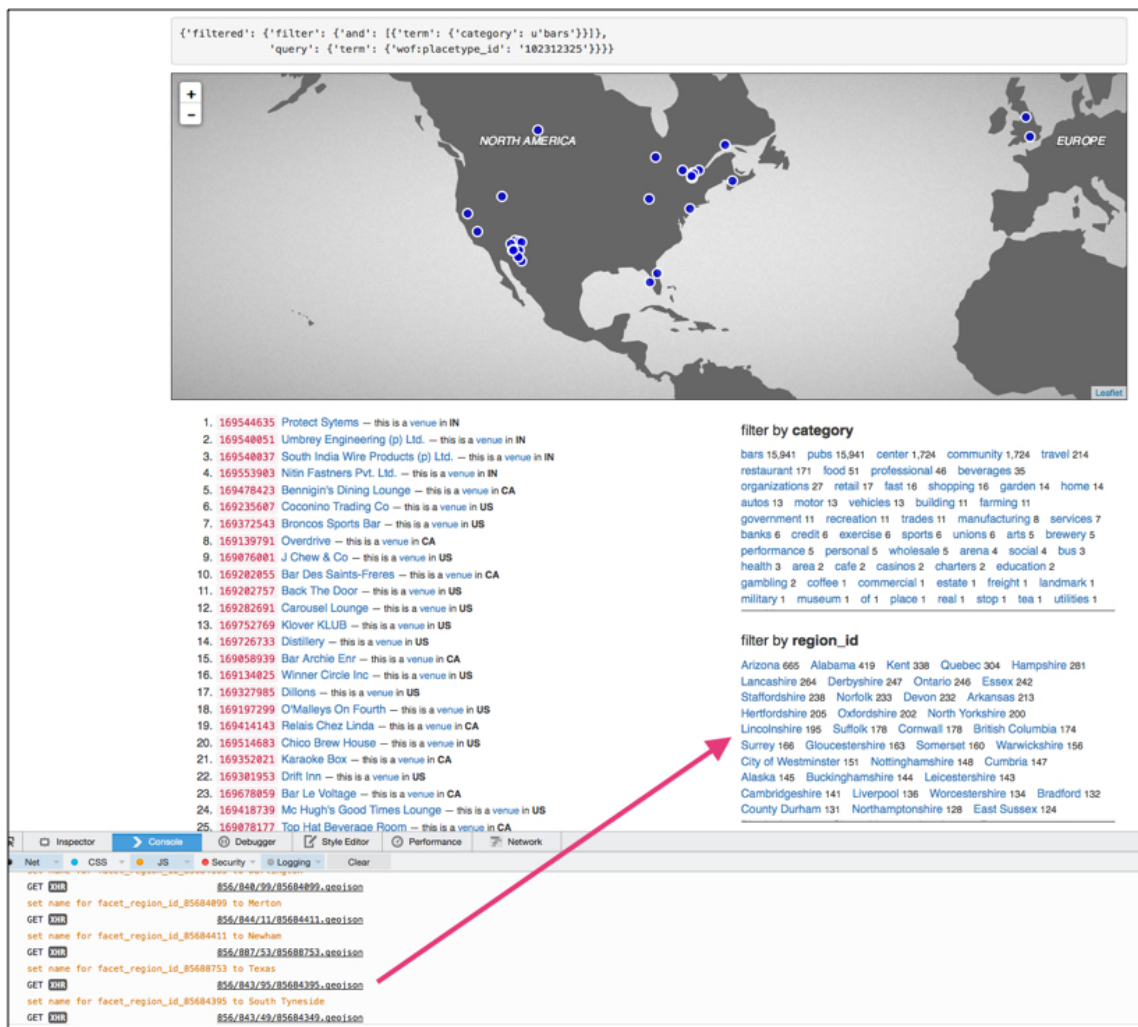
Here's a screenshot of **the state of California**

(<http://whosonfirst.mapzen.com/spelunker/id/85688637/>), the pink shape in the middle of the screen. The large yellow shape behind it is **the United States** (<http://whosonfirst.mapzen.com/spelunker/id/85633793/>).



What's happening here is that only the metadata (or `properties` hash) for California is initially sent to the web browser. The metadata contains both a unique ID and a pointer (ID) to its parent. Once the web page has loaded there is a little piece of Javascript that generates the URLs for each location's complete information including their geometries which are then drawn by the browser.

A practical reason for doing this is that often the geometries for countries and regions are large and prohibitive to shuttle from a database to a web server and then finally on to a web browser. A network of documents allows to distribute the cost of rendering a series of complex relationships between the server and the browser.



This is another example of the same idea but fetching pointers to display place names rather than geometries.

What you're seeing is a screenshot of a read-only "spelunker" that we've built for exploring the Who's On First data. The column on the left shows the results for a query of records whose placetype is `venue` and whose category is `bars`. The column on the right shows the same results grouped by a particular facet; all the distinct categories or regions contained in the result set and so on.

Because we store only pointers to the other locations associated with a place faceting by region returns a whole bunch of numeric IDs. Employing the same trick we use to draw geometries we can use those pointers to fetch their corresponding records, from the network, and draw their place names on the fly.

ALL TEH SPELLINGS

And this is important because one of the things that a gazetteer (and stable identifiers) allows you do is: Include *all the names* for a place. All the names, all the variations, all the translations.

ALL THE GEOMS

The same principle applies to geometries for a location. We are used to thinking that there is only ever one canonical geometry (or polygon) for a place but in practice that's not entirely true.

For example, there might a complex **ground-truth** (https://en.wikipedia.org/wiki/Coastline_paradox) style geometry which is very detailed and contains all the nooks and crannies that define the contour of a place. This geometry will likely be very large and sometimes all you need is a display geometry to show a small map on a mobile phone. There is no need for anything as large as a ground-truth geometry. Likewise if you're doing reverse-geocoding you might want a geometry that includes a country's territorial waters that extend beyond the coastline. The list goes on.

concordances

We also store as many concordances to other gazetteers as possible. We want Who's On First to be as thorough and comprehensive of a dataset as we can make it but that doesn't mean we need to or want to **drink everyone else's milkshake** (https://en.wikipedia.org/wiki/I_drink_your_milkshake#In_popular_culture). We would rather store pointers to the work that others are doing.

relationships (all families are psychotic)

Likewise relationships and hierarchies. More specifically, we have chosen to allow places to have *multiple* hierarchies because that simply reflects the reality of the place or placetype.

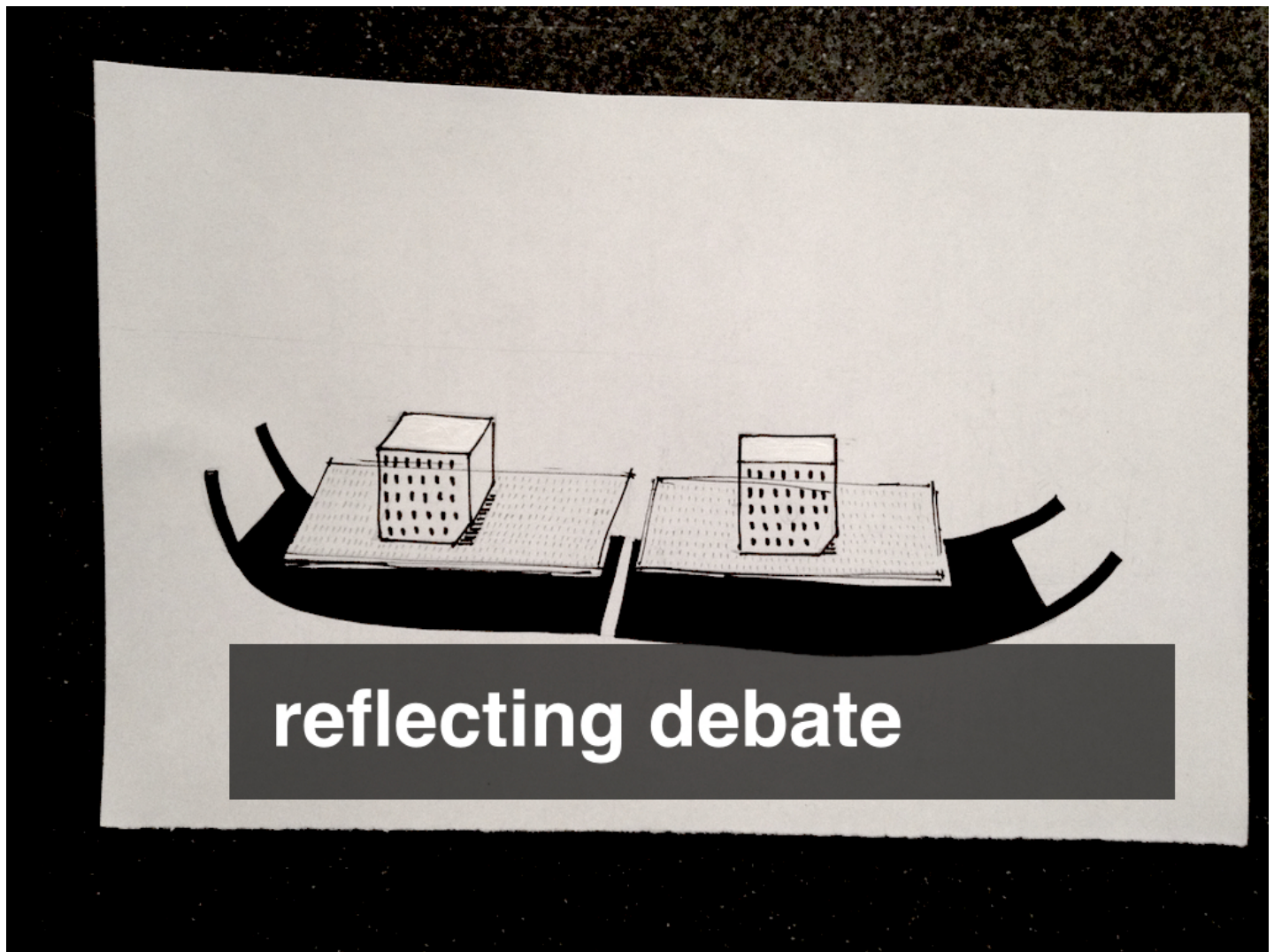
Consider the **“Bay Area”** (https://en.wikipedia.org/wiki/San_Francisco_Bay_Area) in and around the city of San Francisco. It and many other **metropolitan areas** (<https://github.com/straup/whereonearth-metropolitan-area>) like it are typically parented by at least two or more counties. Consider a land disputed by Israel and Syria or China and India or India and Pakistan. The list goes on.



supersedes (superseded by)

The last thing I want to mention before moving on are two properties that every record contain: `supersedes` and `superseded_by`. Neither property attempts to encode the semantics of *why* or *how* a place might have been superseded only that it has been.

These properties allow us reflect the change that a place has undergone while still preserving a stable record of the place it used to be. For example, Yugoslavia which existed in three distinct forms during the 20th century before finally dissolving in the 1990s.



Hopefully you're starting to see a theme. Who's On First is not an attempt to settle debates about place but rather to provide a scaffolding that can reflect the debate about a place and allow for as many decisions about how assertions about a place are interpreted.

“spelunking”

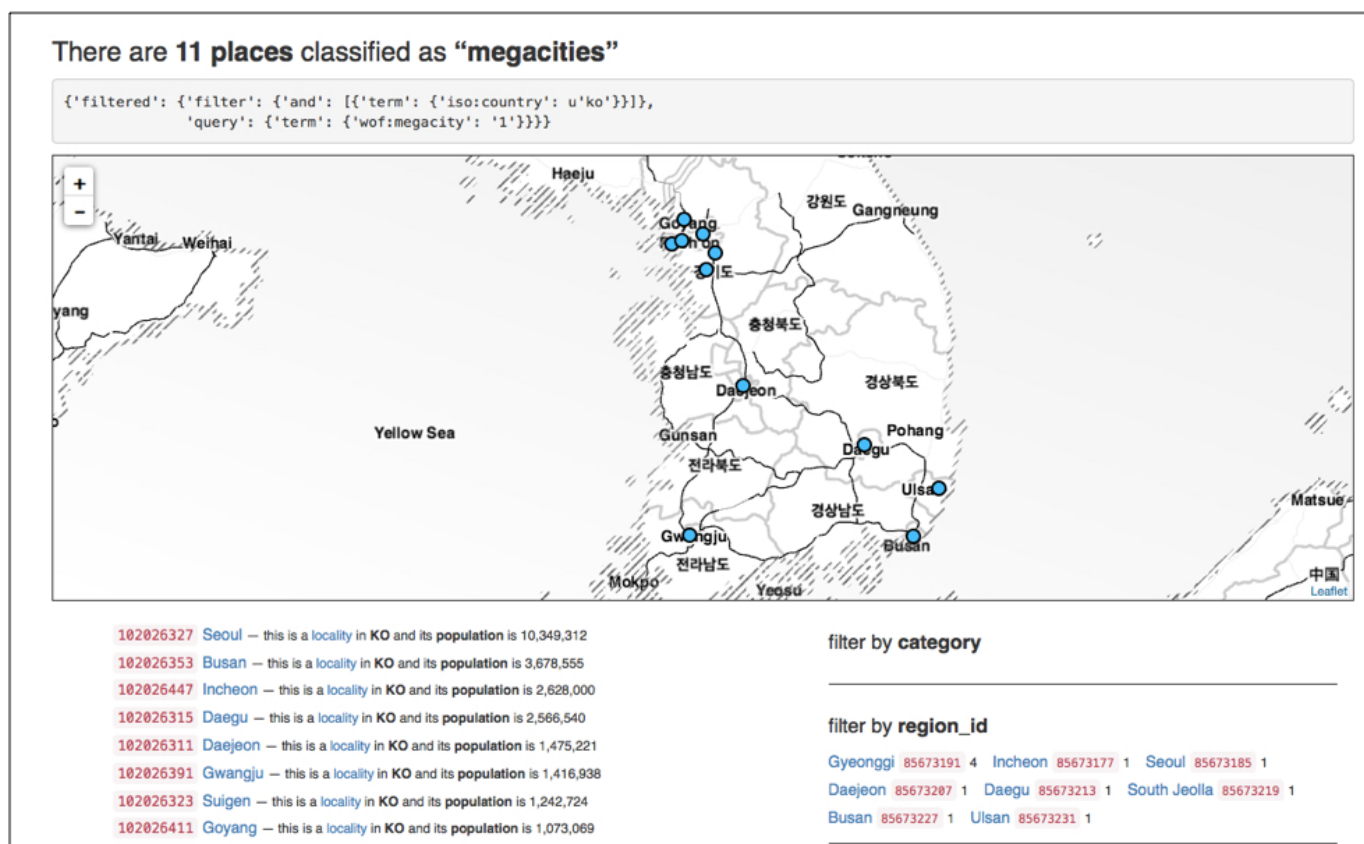
I've mentioned something called a spelunker, a few times now.

Spelunking (<https://www.wordnik.com/words/spelunking>) is an expression used to describe the act of exploring caves, of feeling your way around an unknown (and often dark) space **by instinct and intuition** (<https://goodformandspectacle.wordpress.com/tag/spelunker/>).

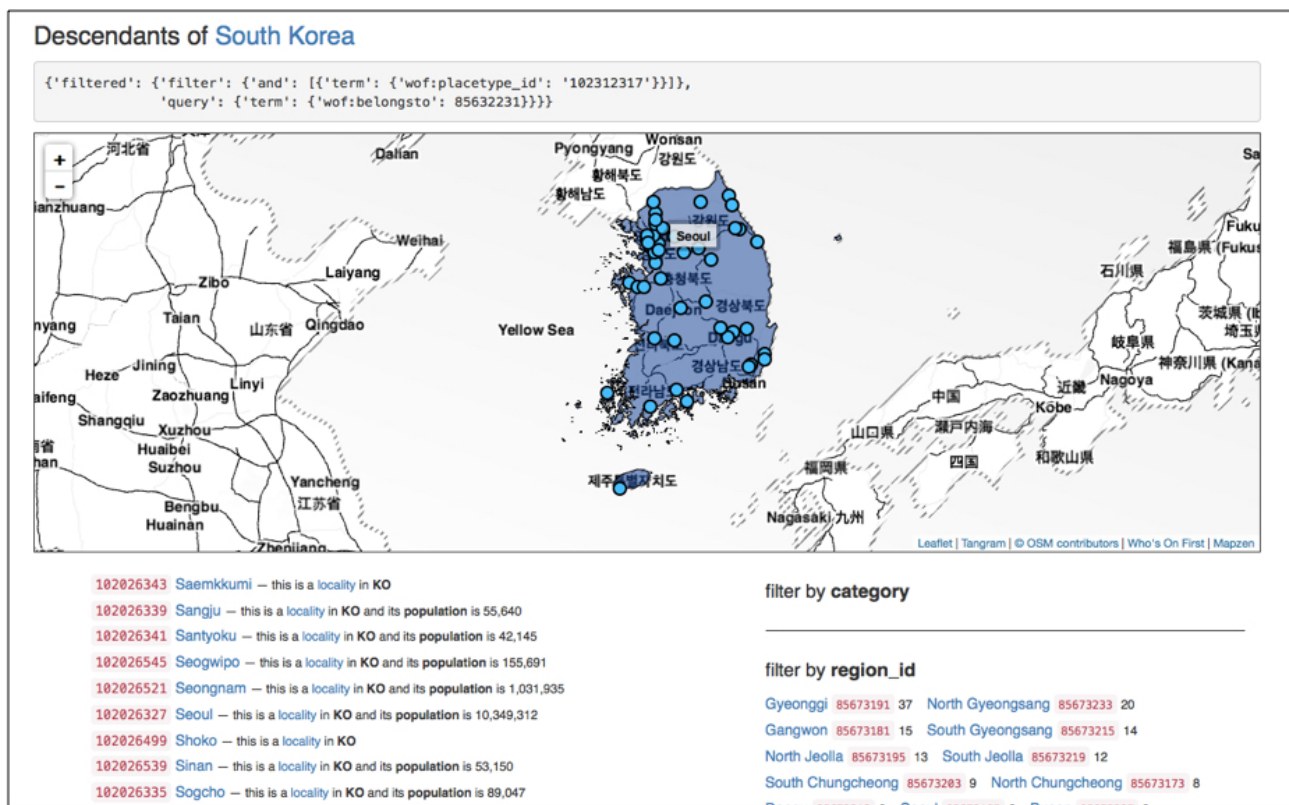
Dealing with large datasets is often the same and Who's On First is nothing if not a large dataset.

The spelunker is the name we've given to **a little read-only web-application** (<http://whosonfirst.mapzen.com/spelunker/>) we've written for searching and browsing and sanity checking all the data we've imported so far.

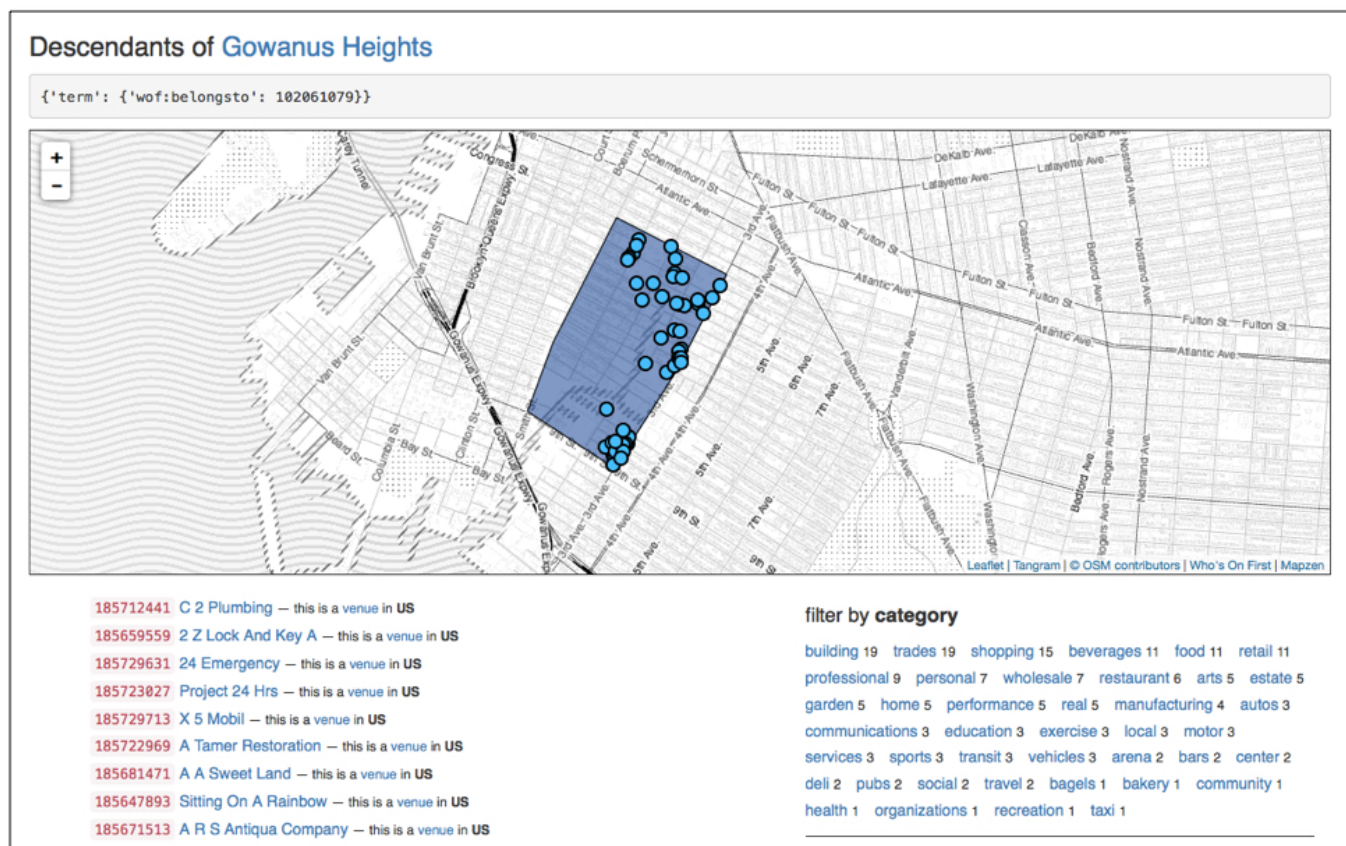
All the Who's On First data is indexed in both **Elasticsearch** (<https://www.elastic.co/products/elasticsearch>) (without geometries) and **PostGIS** (<http://postgis.net/>) (without properties). Currently the only functionality in the spelunker is available through the Elasticsearch endpoint but that will evolve soon enough. I mentioned that the goal is to ensure that Who's On First is not particular to any one database and that is part of the exercise with the spelunker.



Here are all the localities in South Korea that have been **classified as “megacities”**
 (<http://whosonfirst.mapzen.com/spelunker/megacities/?&iso=ko>).

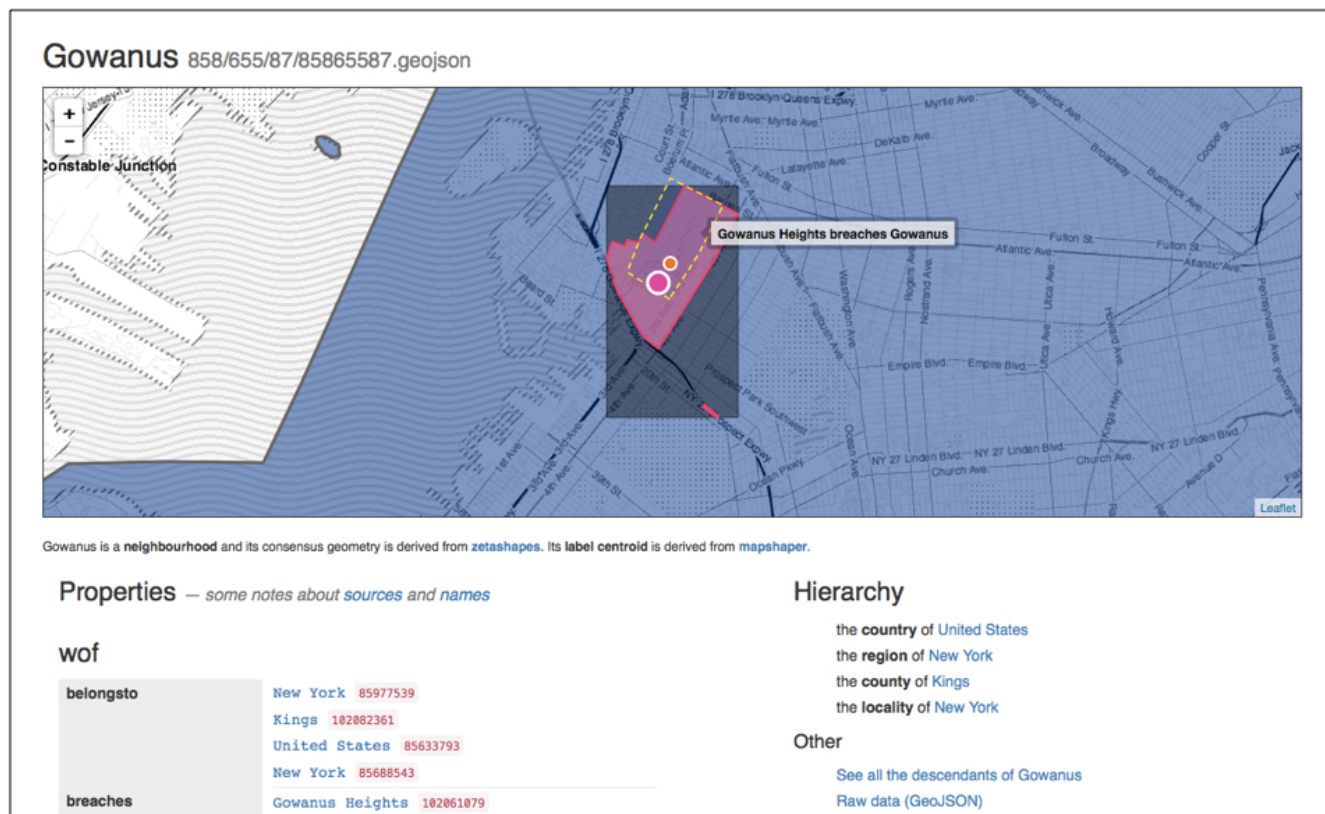


Here are all the **localities that are descendants of South Korea**
 (<http://whosonfirst.mapzen.com/spelunker/id/85632231/descendants/?&placetype=locality>).



Here is **the neighbourhood of Gowan Heights**

(<http://whosonfirst.mapzen.com/spelunker/id/102061079/>) in Brooklyn, New York. Gowan Heights has a little bit of a contested history and **its creation** (<http://www.aaronland.info/weblog/2013/02/03/reality/#youarehere>) is a story best saved for another talk.



I mention it to demonstrate one of the properties of the Who's On First data and the spelunker that we've started to work on: Breaches.

Breaches are geometries of the same placetype that overlap one another.

Gowanus Heights "breaches" the neighbourhood of **Gowanus** (<http://whosonfirst.mapzen.com/spelunker/id/85865587/>). We've only encoded breaches for a handful of locations so far but the plan is to keep track of all the breaches for all the places.

We want to do this for two reasons: First, they can serve as internal flags for records that still need to be vetted or edited. Secondly they can act as signals to people using the Who's On First data that a given location might still be open to interpretation.

All neighbourhoods are disputed, after all.

Gowanus 858/655/87/85865587.geojson

Gowanus is a neighbourhood and its consensus geometry is derived from [zetashapes](#). Its label centroid is derived from [mapshaper](#).

Properties — some notes about sources and names

wof

belongs to

New York 85977539

Kings 102082361

United States 85633793

New York 85688543

breaches

Gowanus Heights 102061079

Hierarchy

the country of United States

the region of New York

the county of Kings

the locality of New York

Other

See all the descendants of Gowanus

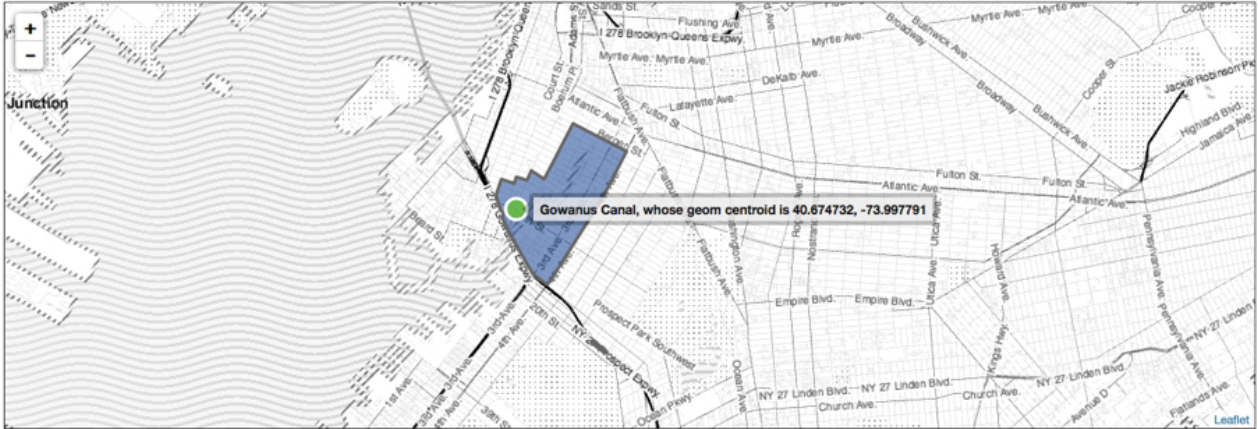
Raw data (GeoJSON)

You may have noticed that there are two circular markers contained by the geometry for Gowanus. One of them is the arithmetic centroid for that geometry and the other is the label centroid – where the best place to display the name for this location, as determined by Matthew Bloch’s excellent **Mapshaper** (<https://github.com/mbloch/mapshaper>).

<https://mapzen.com/blog/spelunker-jumping-into-who-s-on-first/>

26/37

Gowanus Canal 185/662/657/185662657.geojson



Gowanus Canal, whose geom centroid is 40.674732, -73.997791

Gowanus Canal is a **venue** and its consensus geometry is derived from [simplegeo](#).

Properties — some notes about [sources](#) and [names](#)

wof

| belongsto | name | id |
|---------------|-----------|----|
| Gowanus | 85865587 | |
| United States | 85633793 | |
| New York | 85977539 | |
| Kings | 102082361 | |
| New York | 85688543 | |

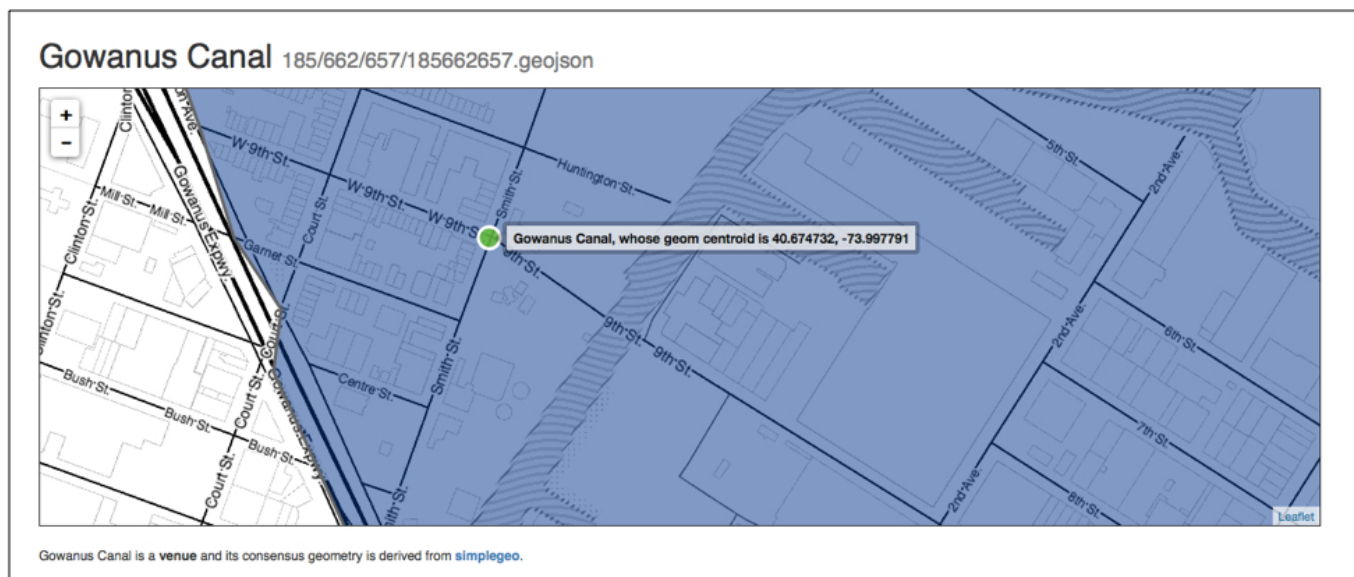
Hierarchy

- the [neighbourhood](#) of Gowanus
- the [locality](#) of New York
- a [continent](#) that we aren't able to index correctly because... ?
- the [country](#) of United States
- the [region](#) of New York
- the [county](#) of Kings

Other

Here is **the Gowanus Canal** (<http://whosonfirst.mapzen.com/spelunker/id/185662657/>) which is parented by the neighbourhood of Gowanus.

It was classified as a **“venue”** (<http://whosonfirst.mapzen.com/spelunker/placetypes/venue/>) by SimpleGeo who published a CC0 dataset of business listings in 2010 (<https://pinboard.in/u:mapzen/t:simplegeo>) and which we are in **the process of importing** (<https://github.com/whosonfirst/whosonfirst-venue>).

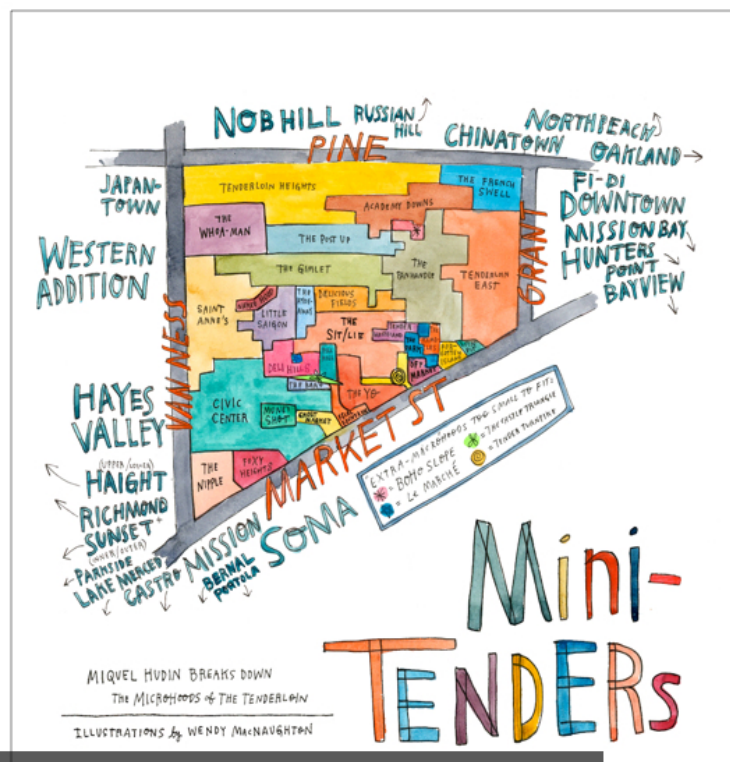


If you're not from New York you may not appreciate just how wrong the current data for the Gowanus Canal is.

Aside from the fact that there is a Korean taco joint at the corner of 9th and Smith now, **the Gowanus Canal is literally a toxic waste site (<http://www.gowanuscanal.org/>)** and when the canal floods it is generally understood to be a public health emergency.

This sort of discrepancy is exactly what the spelunker was built to uncover.

And yes, you read that right: A toxic waste site in the middle of Brooklyn does in fact flood in to the streets, from time to time...



ground truthiness

The spelunker is also a way for us to see discontinuities between administrative perspectives and the realities on the ground.

In 2011 **Miquel Hudin** (<https://www.hudin.com/>) and **Wendy MacNaughton** (<http://wendymacnaughton.com/>) published a piece, in the online review *The Bold Italic*, about the often derided and more frequently neglected **San Francisco neighbourhood of the Tenderloin** (<http://www.thebolditalic.com/articles/1101-mini-tenders>). Hudin wrote:

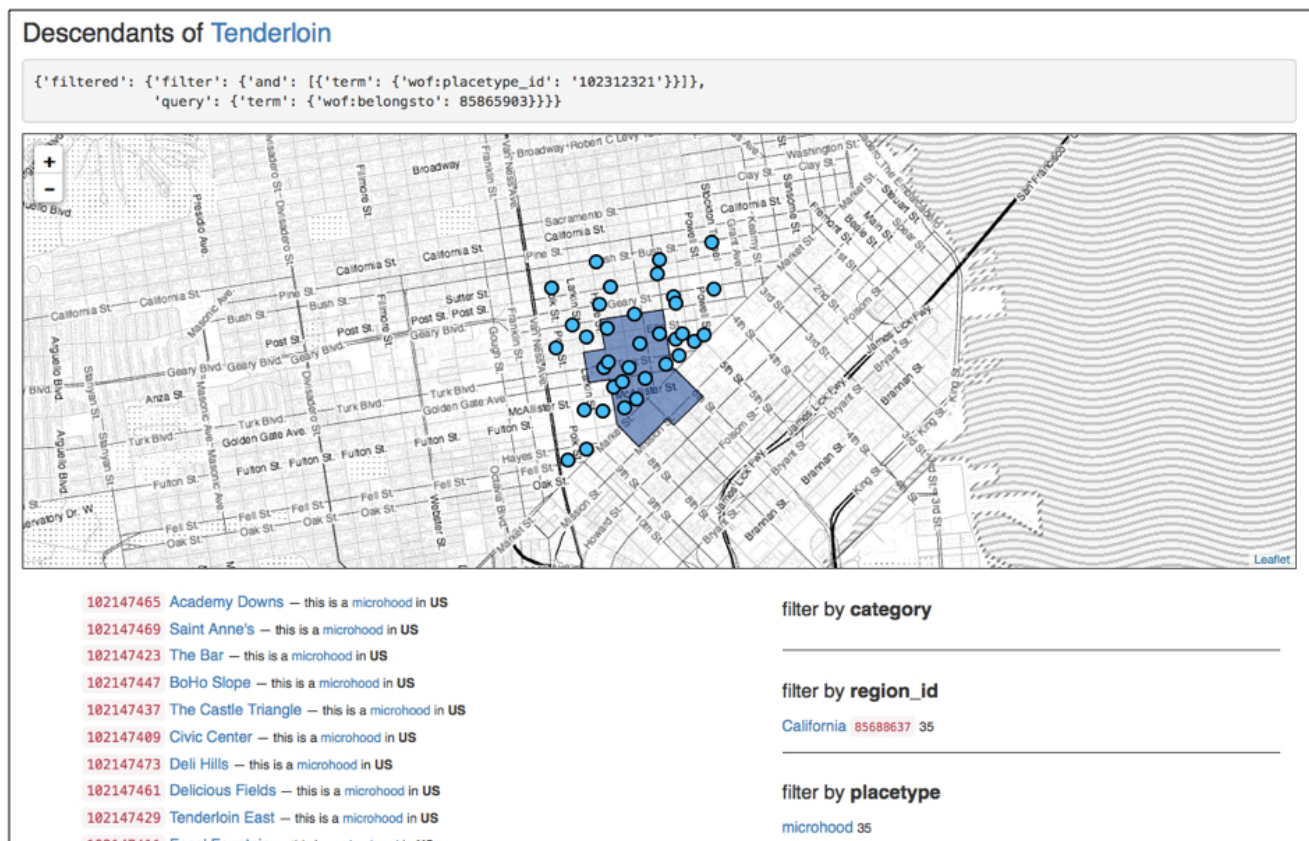
Most consider our Tenderloin neighborhood to be a vast black hole of no-go in downtown San Francisco.

Truth be told, there are certainly a couple of blocks full of downright nasty that neither you nor I should make a habit of frequenting. But beyond those unfortunate social potholes, the Tenderloin is a rich neighborhood with a great wealth of small areas each with their own character.

Since The Bold Italic popularized the term “microhood,” it’s only fitting to break down the Tenderloin by the sum of its parts. So, presented here is the “Tenderloin Microhoods Map.” While some of it is just for fun, a whole lot of it is most definitely true.

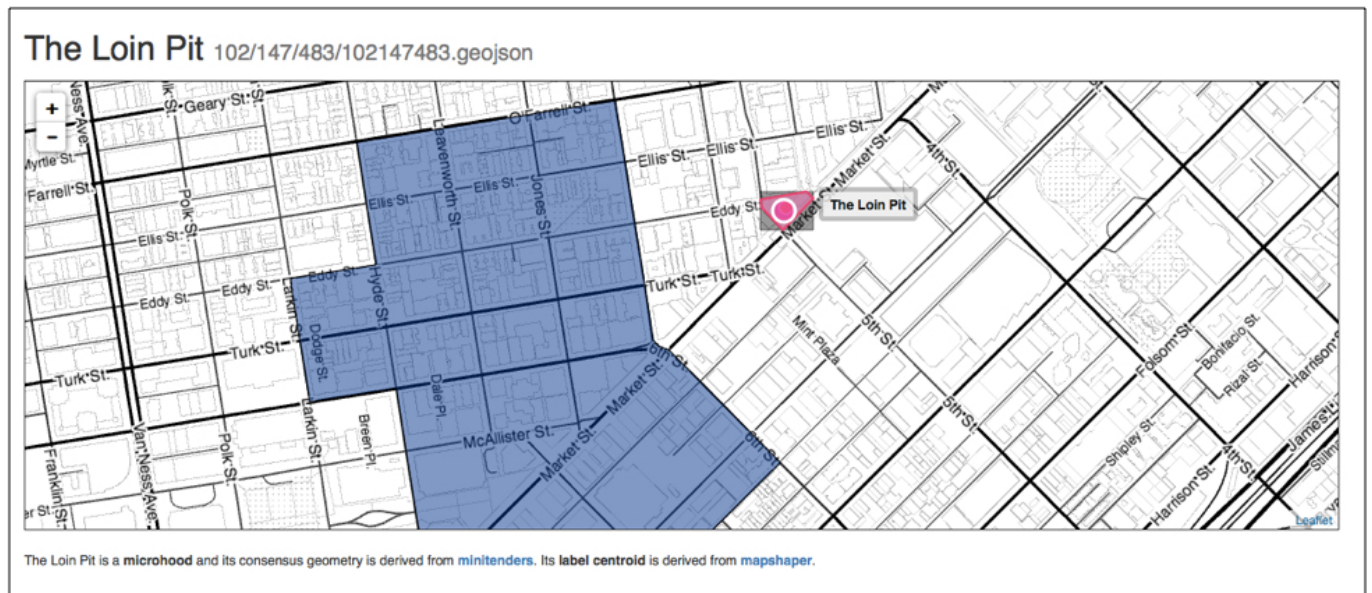
So, in 2015 we imported all those microhoods into Who’s On First, **classified them accordingly** (<http://whosonfirst.mapzen.com/spelunker/placetypes/microhood/>) and parented them by the **Tenderloin** (<http://whosonfirst.mapzen.com/spelunker/id/85865903/>).

Rather than do a spatial point-in-polygon test to ensure that each on of the microhoods was contained by the geometry for the Tenderloin we simply assigned them a parent ID of **85865903** (<http://whosonfirst.mapzen.com/spelunker/id/85865903/>).



And here's what that looks like.

The geometry for the Tenderloin was sourced from David Blackman's **Zetashapes** (<http://www.zetashapes.com/>) project, derived from a number of different sources. The geometries for the Mini-Tenders were sourced by a pair of local San Franciscans.



This is just a detailed view of the last slide. Here the microhood called **The Loin Pit** (<http://whosonfirst.mapzen.com/spelunker/id/102147483/>) sits entirely outside the geometry of its parent neighbourhood, the **Tenderloin** (<http://whosonfirst.mapzen.com/spelunker/id/85865903/descendants/>).

The issue right now isn't so much that one or the other is right (or at least more correct) but that we have a way to see those differences.

Properties
— some notes about sources and names
make pretty

```

# This is the raw properties hash from the source data itself.
# It _should_ magically transform itself in to a pretty formatted
# table and if it doesn't that probably means there's something wrong
# with the data itself (or maybe it just hasn't been synced yet).
# Or maybe you pressed the "view raw" button to see the raw data.
# Because raw data is raw.

{u'geom:area': 0.0,
 u'geom:bbox': u'-122.408546,37.784046,-122.407539,37.784622',
 u'geom:latitude': 37.784386,
 u'geom:longitude': -122.408065,
 u'iso:country': u'US',
 u'lbl:latitude': 37.784335,
 u'lbl:longitude': -122.408101,
 u'mps:latitude': 37.784335,
 u'mps:longitude': -122.408101,
 u'name:eng_p': [u'The Loin Pit'],
 u'src:geom': u'minitenders',
 u'src:geom_alt': [],
 u'src:lbl:centroid': u'mapshaper',
 u'wof:belongsto': [85688637, 85633793, 85922583, 102087579, 85865903],
 u'wof:breaches': [],
 u'wof:concordances': {},
 u'wof:geomhash': u'728378eaf4ea5474ac0453e81718e26f',
 u'wof:hierarchy': [{u'country_id': 85633793,
 u'county_id': 102087579,
 u'locality_id': 85922583,
 u'microhood_id': u'102147483',
 u'neighbourhood_id': 85865903,
 u'region_id': 85688637}],
 u'wof:id': 102147483,
 u'wof:lastmodified': 1441062619,
 u'wof:name': u'The Loin Pit',
 u'wof:parent_id': u'85865903',
 'wof:path': '102/147/483/102147483.geojson',
 u'wof:placetype': u'microhood',
 u'wof:placetype_id': 102312321,

```

By default the spelunker displays the properties for each record as a raw blob of JSON. Once the page has loaded there is code that (again) fetches the record for the current location over the network, parses the properties data and renders the same data as an HTML table.

If this approach sounds both inefficient and like overkill that's because it probably is.

At the time it was also easier than parsing the escaped JSON in the html `<pre>` block (which wasn't working because... computers?) and allowed to further test the idea of fetching, parsing and displaying remote data after the page had loaded.

In some ways the spelunker has been specifically designed to be potentially inefficient as a way to help us identify the stress points in the models we've chosen for Who's On First.

Properties — some notes about [sources](#) and [names](#)

[view raw](#)

wof

| | | | | | | | | | | | | | |
|------------------|---|--------------|-----------|-----------|--------------------|------------|-----------------------|-------------|-----------------------|-----------|------------------------|------------------|--------------------|
| belongsto | <div>California85688637</div> <div>United States85633793</div> <div>San Francisco85922583</div> <div>San Francisco102087579</div> <div>Tenderloin85865903</div> | | | | | | | | | | | | |
| breaches | — | | | | | | | | | | | | |
| concordances | | | | | | | | | | | | | |
| geomhash | 728378eaf4ea5474ac0453e81718e26f | | | | | | | | | | | | |
| hierarchy | <table><tr><td>microhood_id</td><td>102147483</td></tr><tr><td>region_id</td><td>California85688637</td></tr><tr><td>country_id</td><td>United States85633793</td></tr><tr><td>locality_id</td><td>San Francisco85922583</td></tr><tr><td>county_id</td><td>San Francisco102087579</td></tr><tr><td>neighbourhood_id</td><td>Tenderloin85865903</td></tr></table> | microhood_id | 102147483 | region_id | California85688637 | country_id | United States85633793 | locality_id | San Francisco85922583 | county_id | San Francisco102087579 | neighbourhood_id | Tenderloin85865903 |
| microhood_id | 102147483 | | | | | | | | | | | | |
| region_id | California85688637 | | | | | | | | | | | | |
| country_id | United States85633793 | | | | | | | | | | | | |
| locality_id | San Francisco85922583 | | | | | | | | | | | | |
| county_id | San Francisco102087579 | | | | | | | | | | | | |
| neighbourhood_id | Tenderloin85865903 | | | | | | | | | | | | |
| id | 102147483 | | | | | | | | | | | | |
| lastmodified | 2015-08-31T23:10:19.000Z | | | | | | | | | | | | |
| name | The Loin Pit | | | | | | | | | | | | |
| parent_id | Tenderloin85865903 | | | | | | | | | | | | |
| placetype | microhood | | | | | | | | | | | | |
| superseded_by | — | | | | | | | | | | | | |
| supersedes | — | | | | | | | | | | | | |

This is what the same data looks like rendered as a series of nested tables. See the way the raw properties blob only contains pointers (IDs) and the pretty version contains names? More network requests! Requests are cached on the page, though, so we only fetch the record for an individual location once.

It's also worth considering caching the same data locally (using some form of HTML5 local storage technology) but we haven't done that yet.

The spelunker is a read-only application, by design. That said we are planning to build a tool for editing Who's On First documents and the value of the spelunker is to help us *start* to think through the interface details of how an editing tool should work.

We are starting with a read-only interface as a way to understand what parts of that interface should be dynamic. Editing is as much about review and scanning, looking for errors or inconsistencies, so that's the first step.

theory:
www.mapzen.com /
blog /
who-s-on-first

This has been a short talk and I have only touched on some of the highlights of the Who's On First project. Location is a complicated subject and it's early days still but there is a long and detailed blog post discussing the motivations for shouldering the burdens of a project like this:

<https://mapzen.com/blog/who-s-on-first> (<https://mapzen.com/blog/who-s-on-first>)

practice:
whosonfirst.mapzen.com /
spelunker /

The spelunker itself is available for... spelunking at:

<http://whosonfirst.mapzen.com/spelunker/> (<http://whosonfirst.mapzen.com/spelunker/>)

Like everything we do at Mapzen **the spelunker is an open-source project** (<https://github.com/whosonfirst/whosonfirst-www-spelunker>) and available for review and download on GitHub.

Thank you.

· 28 September 2015 ·



Aaron Straup Cope

Aaron is happiest in the presence of olive oil.

© 2017 Mapzen

This is It: Mapzen Search is now live

`search` (`/tag/search`)

Mapzen Search (<https://mapzen.com/projects/search>) is our modern geographic search service based entirely on **open-source tools** (<https://github.com/pelias>) and powered by entirely open data. And as of today it's **open for all to use** (<https://mapzen.com/projects/search>).

With Mapzen Search you can transform the natural language that we use to describe places in to geographic coordinates that computer systems can understand. You can also do the opposite and turn a latitude and longitude values in to a list of nearby places.

You can **sign up for an API key** (<https://mapzen.com/developers>) and start building with it today.

Out of the gate, it comes with data from some of the greatest open data sources on the planet:

- **OpenStreetMap** (<http://openstreetmap.org>)
- **OpenAddresses** (<http://openaddresses.io>)
- **Quattroshapes** (<http://quattroshapes.com/>)
- **Geonames** (<http://geonames.org>)

And in the next few weeks, we'll be turning on our integration with **Who's on First** (<http://whosonfirst.mapzen.com>), Mapzen's global gazetteer and rolling out improvements across the board. It's an exciting place to be.

How We Got To Now

Searching the world is **a pretty hard problem** (<https://mapzen.com/blog/the-world-is-yours-announcing-mapzen-search>), and one we've been working on at Mapzen from close to our start.

On November 13, 2013 Randy Meech, our CEO, **committed the first version of Pelias** (<https://github.com/pelias/posterity/commit/183ea89bf5380ff6ebee0fd8bae472fb673978c8>), our open source geocoder to Git, noting `USAGE: This is experimental!`¹ But the code got its start several weeks earlier, during the 2013 Code For America Summit. It was a fitting place to start writing a geocoder and an apt way to frame what we were about to set out to do. It's also fitting that our Engineering Director for Search, Diana Shkolnikov is **launching Mapzen Search at the 2015 Code for America Summit today** (https://www.codeforamerica.org/summit/schedule/#breakout_133)!

From its start, Pelias has had 3 core values:

- It had to be open source, so people could be free to use it on their own, but also to use it as the basis of their own geocoding and place search services
- It had to work on top of open data, but be fundamentally data agnostic, so if others chose to run it, they could do so with their own data
- And most importantly, it had to have a broad, global contributor base that was willing to share their knowledge of how places were organized across the world, so we could best let people represent where they live and to adapt to the inevitability of places changing over time

Mapzen Search continues to build on that core ideology. It's run entirely as a stock version of our modern version of **Pelias** (<https://github.com/pelias>) and powered entirely by open data.

We've been continually amazed by our community of contributors, Whether you've been using Mapzen Search while in public beta for the past year², been **running Pelias** (<https://github.com/pelias/vagrant>) on your own servers, or sharing data and code, you've helped create an open infrastructure that makes it possible for us to see and understand the world around us.

It is an absolute privilege for us to build Mapzen Search with so many of you. Use it; it's yours.



(<http://digitalcollections.nypl.org/items/b631f3c8-eb18-6be6-e040-e00a18064c28>)

Hunt-Lenox Globe (https://en.wikipedia.org/wiki/Hunt-Lenox_Globe) - Rare Book Division, The New York Public Library. "Full view of globe with stand." *The New York Public Library Digital Collections*. 1512. <http://digitalcollections.nypl.org/items/b631f3c8-eb18-6be6-e040-e00a18064c28> (<http://digitalcollections.nypl.org/items/b631f3c8-eb18-6be6-e040-e00a18064c28>)

1. Pelias wasn't the first piece of Mapzen code. Our other co-founder Brett Camper started work on **Tangram** (<https://github.com/tangrams>) on **September 29, 2013** (<https://github.com/tangrams/tangram/commit/dc706b1c8dab198de51521860d7feec379f6a3d7>), then called Canvas Map. ↩
2. We'll be sharing more about some of the remarkable things folks have already done using Mapzen Search over the next few weeks. ↩

· 30 September 2015 ·



David Riordan

Former product manager for Mapzen Search. Historical mapping, open tools, open access, and open data.

© 2017 Mapzen

Scenes from a SIGGRAPH

tangram (/tag/tangram) demo (/tag/demo)



[Leaflet](#) | [Tangram](#) | © OSM contributors | [Mapzen](#)

[Click maps to interact]

I was recently in Los Angeles for SIGGRAPH, “the world’s largest computer graphics conference.”

I took some notes.



The lyft pulls up in front of a pale-green multi-story warehouse with lots of windows propped open. The street is empty, except for a white-label food truck, the kind called a “lonchera,” parked outside the front door. Inside, the loft is cavernous, like a movie set, where the height of the ceiling is intended to symbolize moral excess.

The airbnb host cheerfully describes it as “Brooklyn style” before giving us keys and locking us in – we are at a loss. Is there another Brooklyn? Does she mean the reclaimed industrial vibe? Surely she can’t mean the pop-culture bricolage, or all the square feet, or the chalkboard wall with the name of an acapella group written in letters three feet high, left over from a music video shoot.

A poster of Lana Turner glowers over a faux-vintage Coke fridge. I realize it all reminds me of a quirky 90’s teen sitcom, minus a working traffic light. None of the bedrooms have doors. Outside, the lonchera folds up and drives away. Trulia says the elementary school is below average.

Two days later, the lonchera will sell me a fried egg on corner-store wheat bread for two dollars at eight am local time, and it will be delicious.



The airbnb is in the Fashion District, which is to New York’s Garment District as a lemonade stand is to Wall Street. We walk to the convention center past a steady stream of people setting up racks of clothes outside of what appears to be a never-ending outlet mall with signs from the 80’s. Norteño plays from actual battery-powered portable radios. As far as I can tell, I never see a single customer on the street the whole time I am in town. Maybe it’s all a front of some kind. Or a set for a chase scene through a crowded marketplace. Maybe all the stores sell to each other, like the potlatch system of the Pacific Northwest, where the real currency is social capital. I take a picture of a store named “Listicle,” but the light is so bad I can’t bear to tweet it.



The Los Angeles Convention Center is the size of an airport, and reeks of low-budget sci-fi student film, full of odd angles, industrial pastels, and greenish-blue infrared-absorbing glass. It inspires odd behavior until you acclimate to the scale of the place. I panic at the size of the line for the Aardman talk and jump the queue right as they open the doors. No one notices. Who line-jumps at SIGGRAPH? It’s 82 degrees outside, all the palm trees are imported, and the auditorium seats a thousand. I try to relax.



Aardman Animations is an English stop-motion animation studio, and has been around for ages. The talk is a family history of the studio, complete with recently-unearthed home videos of people in unironically-large eyeglasses and earnest expressions painstakingly doing tedious things with electronics. I am most impressed by the way they hack their film cameras to expose single frames, with a video camera patched into the viewfinder so they can see the shot on tv. The moment in the presentation when they move to all-digital feels as clunky as it did at the time, and still looks just like 3D student work – drunk with power, they overuse short lenses and crazy camera moves, and it takes a while for their style to settle back out.

They pronounce it “Aard-Man”, like “Bat-Man”, and not “Aard-mun” as I’ve pronounced it forever. I take a selfie with Wallace and Gromit and tweet it. I learn later that no photos are allowed in any session, or basically anywhere in the building, but nobody said anything. Nobody ever does. I take more pictures, record whole minutes of video. Is this what being from the East Coast is like? Can I do whatever I want if I can stand silent disapprobation?

We go out for Mexican across the street. The restaurant looks like somebody stuffed a mission inside a millionaire’s ranch house. A drink is set on fire. A mariachi trio plays. I can see twelve tvs without turning my head.



The real-time and fake-time rendering worlds are collapsing. This has been foretold. The rendering and compositing techniques make their way from the movie studios to the game studios to the phones, although the whispers say Moore’s Law is stalled out. Even lots of the film sessions are about using real-time techniques – everybody wants stuff faster. We meet the head of the WebGL standards body. He is strikingly congenial. I see the guy who invented bump maps wandering alone through the halls in rope sandals with his lanyard in his beard, like Gandalf. I watch twelve real-time rendering techniques talks in a row, and count at least six involving signed distance fields. One stood out: The Cloud Talk. I’ve been thinking about it for weeks now.



Clouds are beautiful but deadly. I have personally spent dozens of non-billable hours attempting to get clouds to look right. The Cloud Talk brings a series of elegantly-argued lighting equations to play in a narrative which doesn’t hide its dead ends, but which results in some old techniques combined into new ones, replete with shyly-suggested names for the combination. The guy doing the presentation is so “natural philosopher proposinge a neue theorem” I could powder a wig.

Clouds are very difficult to model believably – they bounce light around their guts in unintuitive ways, exhibit wildly different behavior depending on the angle of the sun, cast shadows on their own interiors, and change shape constantly. The Cloud Talk method winds up being half-a-dozen tricks all lined up together, like a Rube Goldberg (Heath Robinson) device running at 60 frames per second, and using signed distance fields.

It's really very good (<https://www.youtube.com/watch?v=bAQpGu29GX4>). It's so good it makes me melancholy.

In Grand Theft Auto, there's a whole Quantified Self screen full of metrics around how you play the game – how many miles driven, run, swam, how many bullets fired, how many minutes spent looking at the sky. I go outside to walk around.



Like most of downtown, Pershing Square is empty. Maybe it's always like that on perfect weekend late afternoons. Maybe Los Angeles is a city of vampires, and they're just asleep. Massive public artworks loom like warnings from a past civilization. Gently decrepit buildings show few signs of habitation – plastic tarps blow in open windows from views which I assume are worth millions, or will be, or used to be. Everything is apparently either frozen in mid-construction or slowly oxidizing. Outside the occasional hip restaurant, people mill like small, brightly-colored fish around a sunken shipping container.



Or maybe the whole city is at the conference. The place is packed. I meet a lot of people who paid their own way here, which surprises me. I meet a very enthusiastic guy who talks at great length about his freelance work, but only in the vaguest of terms. It's later explained to me that he makes interactive ads. I attempt to order a bacon, egg, and cheese sandwich from the cafeteria, but they run out of bacon and cheese so I just have more coffee and watch jpegs fill in over the wifi. In the sessions, there are a lot of survival games being announced, which I find timely, and ominous. I sit on the floor in the back of a dark crowded room and charge my phone in the wall, while distant ghostly blurs from Pixar tell me secrets about geometry lights.



I watch a man give a live demo, right out in public, of a tooth scanner which is basically a toothbrush with a scanner in it. The toothbrush crashes five times, and he has to keep rebooting it, but it finally works. His teeth resolve in pointillistic splendor twenty feet tall on the screen overhead – there is genuine applause and cheering. I imagine he has brushed his teeth very well for the demo. He scans his hand, where he has written his company's URL in blue ballpoint, the ink wicking into the tiny creases in his skin.



An old friend of mine invites me to a virtual reality demo he worked on. I have an intolerance to low latency and poor frame rates and giant pixels but I agree anyway, because virtual reality. The demo is in one section of the massive, darkened, en-curtained exhibit hall called the "Virtual Village" – it's a fenced-off area about the size of a three-car garage, with an overhead lighting grid of pvc pipes, on which are mounted an array of tracking cameras ringed by red LEDs, like night-vision security cameras. Half a dozen people in hoodies mill about inside, looking at a laptop on a folding table, and holding devices up to peer at their undersides. I am dubious, but my friend

declares me a “V.I.P.” so I get to jump the queue. To my face is strapped a Gear VR with a tiny ping-pong tree affixed to the top, and the hoodies velcro four more trees to my wrists and ankles, like the world’s worst forest camo, or antlers on an ill-informed reindeer cosplayer.

The screen is dim. My world is dark. A Wiimote is pressed into my hand, and I’m told that the world is re-launching. And then I am in a cartoon room with three other cartoon stick figures. My Wiimote is a giant cartoon pencil! I draw a box. I draw antlers on the director of the MIT Haptics Lab. I draw a poop emoji, and a giant dinosaur-dragon. I have a good time in virtual reality! It is over very quickly, because there’s a very long line.



At the food trucks, while looking at a map on my phone, I bump into the guy who invented bump maps. I tell this story to at least six people, and then tweet it. I meet the guy who invented Perlin noise, one Mr. Perlin. I tell him I’m a fan of his noise. We get a beer at a bar in a nearby hotel with decor like a 20’s Alhambra, sitting poolside between two massive prickly pear. On three sides of us, above the fences and privacy cacti, the only visible structures are tall buildings under construction. I decide the rest of the city must be somewhere else, the way New York empties out during the holidays. Does LA have Hamptons? Maybe they’re all at the beach.



The diner up the street has a little booth by the front door, like a teller’s window, where you take your check and pay for your meal. There is a shallow bowl-shaped depression in the linoleum in front of the booth from what must be years of people paying, and standing, and shuffling. I count seven layers of linoleum. I presume the hole has been left intentionally – but do they cut a new hole when they put a new layer down? Were there seven when they started? The film industry’s proximity is making me suspicious.



We visit DreamWorks, 30 minutes north by lyft. It is a SoCal movie studio straight out of central casting, with tasteful Spanish colonial architecture and palms everywhere. Everyone working there looks very healthy. Inside, it’s very quiet, the blinds are drawn, and all the ceilings are low and dark, like an empty nightclub at noon. I see secrets, but I’m never sure what I’m looking at – wireframe cartoon heads and reference material. We have lunch under palms in the outdoor cafeteria and talk to multiple Heads of Departments about open data and 3D buildings. They have concerns about licensing. I watch Jeffrey Katzenberg trip over a picnic table leg while carrying a salad around a tiny koi pond. He has a very nice shirt. Everyone calls him “JK.”



I forget how many days I've been here. I regularly get lost in the convention center. I eventually settle into a rat-path between the two main wings, but I never see any pattern in the location of the various types of sessions. I discover that there's another, better coffee shop behind the prominent, bad coffee shop, past the taco stand named "TACOS LA GUERRA", which as far as I can tell means "war tacos" or maybe "taco war." I never try the war tacos, which I now regret.



The main WebGL session is one of the last. It is not in the convention center, but in a Marriott down the street, through a raucous lobby bar, up two escalators, around a corner, and down a very long and empty hallway with a floor-to-ceiling picture-window view of the back of another building. The strikingly congenial standards body head introduces a number of speakers who show WebGL demos.

Graphics cards are, in technical terms, ridiculously powerful. We are in the muscle car era of graphics cards, and we're still figuring out what we can do with them. Most of the standards-body discussion sounds like people trying to figure out where on the rocket-sled it's safe to let people stand. The dominant theme is the eyebrows-up, palms-down "relax, relax, we're all friends here" gesture.

The **Shadertoy** (<http://shadertoy.com>) people are here. The **three.js** (<http://threejs.org>) people are here. API features are announced. Demos are demoed. There are many 3D globes. There's an interactive, interpretative epic poem. There's a unbelievably detailed model of a human heart, animated at the cellular level. Electrical vectors sweep around the heart in whorls of sparking color. Signed distance fields are employed.

I'm too enthralled to take any notes, but I take lots of pictures.



[Leaflet](#) | [Tangram](#) | © OSM contributors | [Mapzen](#)



(Check out the code for these maps at **<http://github.com/meetar/siggraph-maps>**
(**<http://github.com/meetar/siggraph-maps>**))

· 07 October 2015 ·



Peter Richardson

Peter vexes vertices, flusters fragments, and pesters pixels on Mapzen's graphics team.

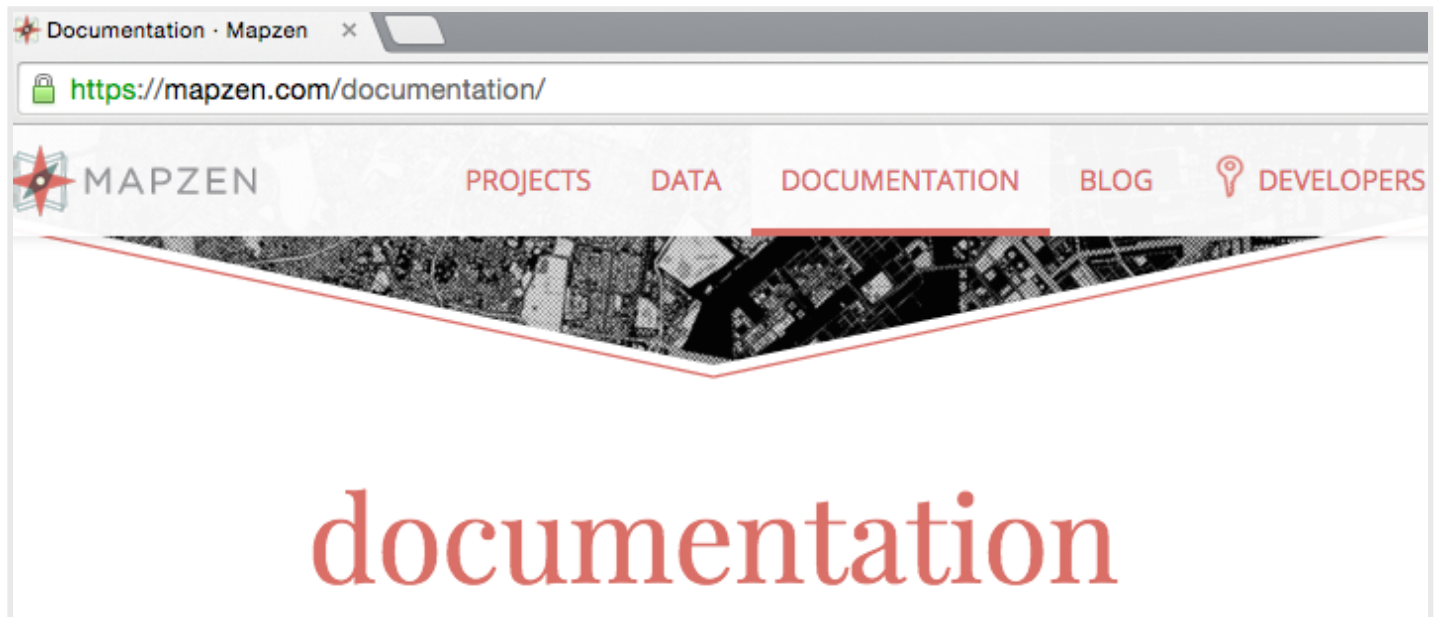
© 2017 Mapzen

Open-source software, open-source documentation

documentation (/tag/documentation)

Mapzen's commitment to open extends beyond software and data—it includes technical documentation, too. We are pleased to introduce a new part of our website that provides a help system built with open-source tools that displays open-source help files.

You can get help at **mapzen.com/documentation** (<https://mapzen.com/documentation/>), or click **Documentation** in the top header on any page on **mapzen.com** (<https://mapzen.com>) to take you there.

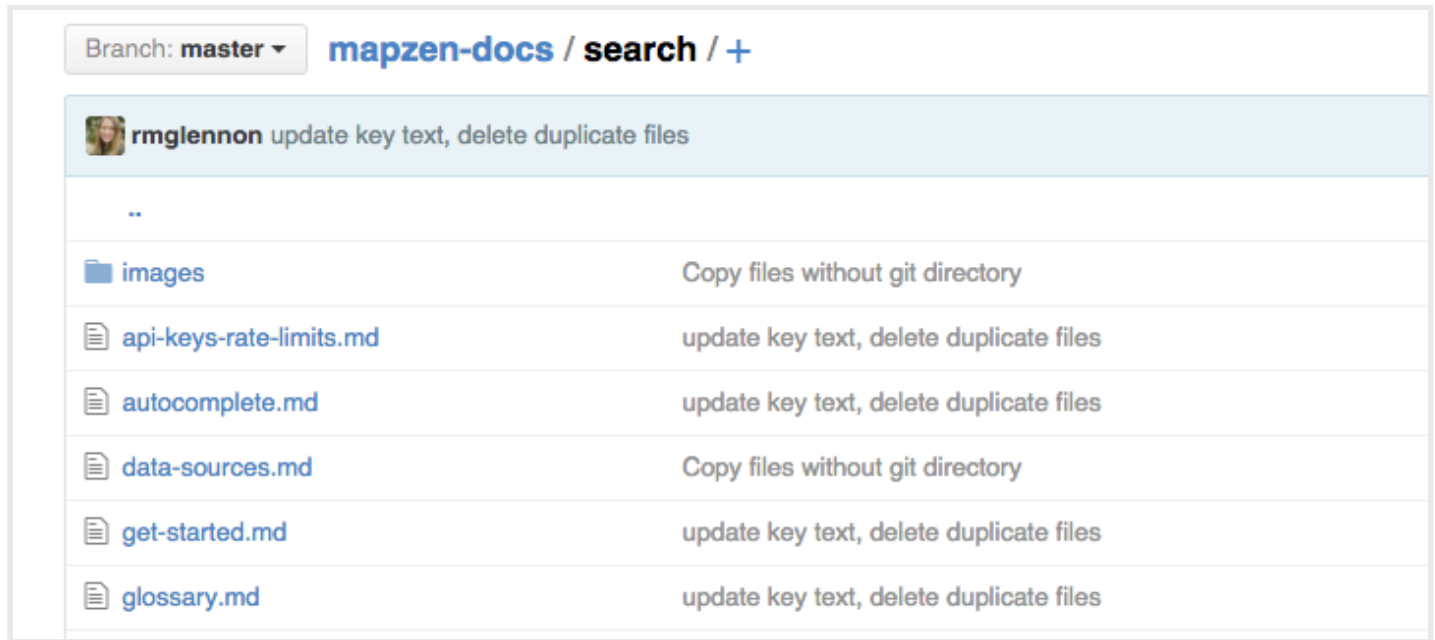


Read on to learn how we built the website and how you can contribute to it.

Setting goals for the documentation site

The main goal of the website is to bring together all Mapzen's documentation in one place. We keep the underlying source help files in GitHub, but display them in a way that is easy to navigate and visually pleasing. In addition, putting the help on mapzen.com connects it to the resources on our website, such as signing up for API keys.

While GitHub is good for storing help files, the contents of a repository is listed by alphabetical order of file names, which may not be the best organization for learning how to use a piece of software. However, with a help system, we can control the table of contents so topics have a readable flow. We can also modify the fonts and source code appearance beyond the defaults on GitHub's website.

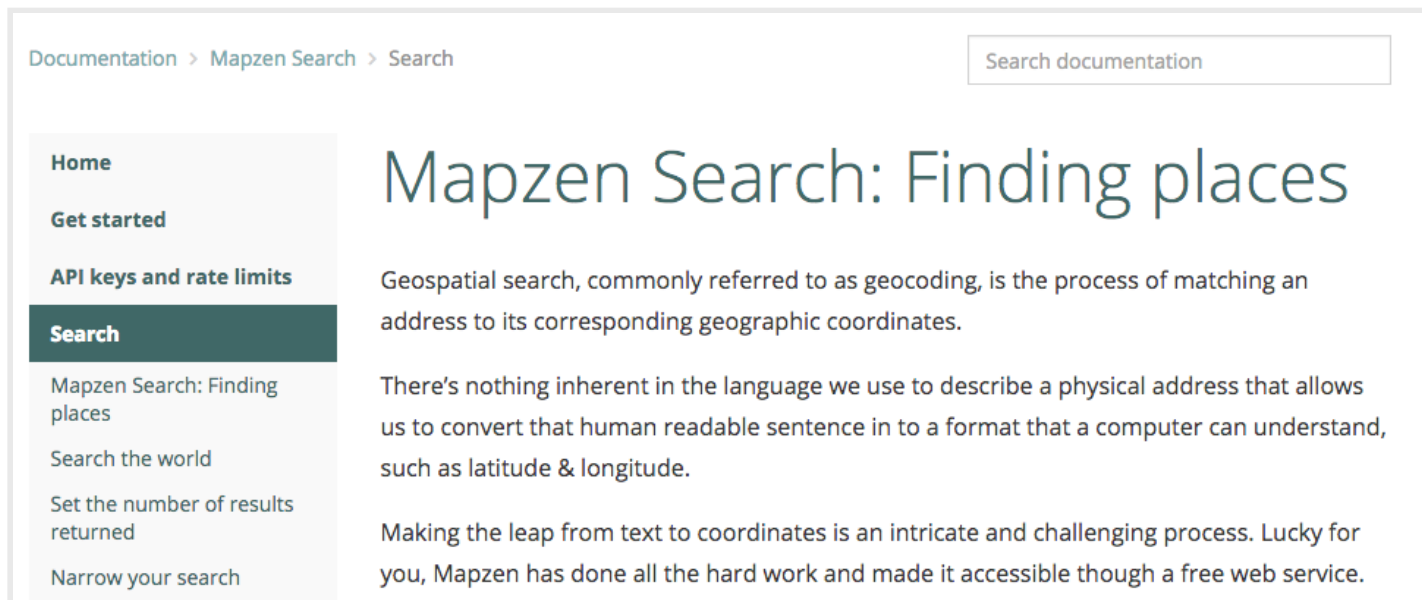


| | |
|---|---|
| Branch: master ▾ mapzen-docs / search / + | |
| rmglennon update key text, delete duplicate files | |
| .. | |
| images | Copy files without git directory |
| api-keys-rate-limits.md | update key text, delete duplicate files |
| autocomplete.md | update key text, delete duplicate files |
| data-sources.md | Copy files without git directory |
| get-started.md | update key text, delete duplicate files |
| glossary.md | update key text, delete duplicate files |

In a repository, files are listed in alphabetical order.

Building the documentation site

After looking at many options for building the help system website, we chose to implement an open-source Python tool called **MkDocs** (<http://www.mkdocs.org/>) to format our GitHub markdown files in to a static, HTML website. MkDocs also creates a table of contents, a simple keyword search, navigation breadcrumbs, and links to move back and forward between topics.



Documentation > Mapzen Search > Search

Search documentation

Home

Get started

API keys and rate limits

Search

Mapzen Search: Finding places

Search the world

Set the number of results returned

Narrow your search

Mapzen Search: Finding places

Geospatial search, commonly referred to as geocoding, is the process of matching an address to its corresponding geographic coordinates.

There's nothing inherent in the language we use to describe a physical address that allows us to convert that human readable sentence in to a format that a computer can understand, such as latitude & longitude.

Making the leap from text to coordinates is an intricate and challenging process. Lucky for you, Mapzen has done all the hard work and made it accessible through a free web service.

In the help system, we can arrange topics in a specified order in the table of contents. The site also has search and navigation functions, along with custom text styling.

The **help site generator** (<https://github.com/mapzen/mapzen-docs-generator>) repository contains the configuration files and visual themes of the resulting site, if you want to contribute or borrow anything for your own documentation projects. Note that while MkDocs reads just one source, our generator can integrate multiple repositories. Because we have added the help system build process to our continuous integration tools, our internal staging website is updated automatically following a change in the source files. When we are ready to deploy the help updates, we pull the changes into the production branch.

Contributing to the documentation

To encourage contributions from users like you, each help topic has an `Edit this page on GitHub` link that points back to the source markdown file. You can then edit a topic and submit a pull request, or add an issue about the content.

We hope that this web site introduces a friendly way of displaying the content and seeing how the topics relate to each other. We'll be continuing to improve the site's design and user experience, as well as enhancing the actual content. We're interested in hearing what you think of Mapzen's documentation and this website. For example, is the help system easy to navigate, are you able to get started and be productive, and can you find what you need? What tutorials or code samples would help you? Send us **feedback** (<mailto:hello@mapzen.com>) or you can log **an issue** (<https://github.com/mapzen/mapzen-docs>).

Beyond this, if you want to help us write great API documentation and build software demos of our products, Mapzen is looking for a **technical writer/developer** (<https://mapzen.com/jobs/technical-writer/>) for our San Francisco office. Send us a **note** (<mailto:jobs@mapzen.com>) if you're interested in joining our team.



This photo was taken from our office window and reflects our view that open-source documentation is another bridge to the community.

· 22 October 2015 ·



Rhonda Glennon

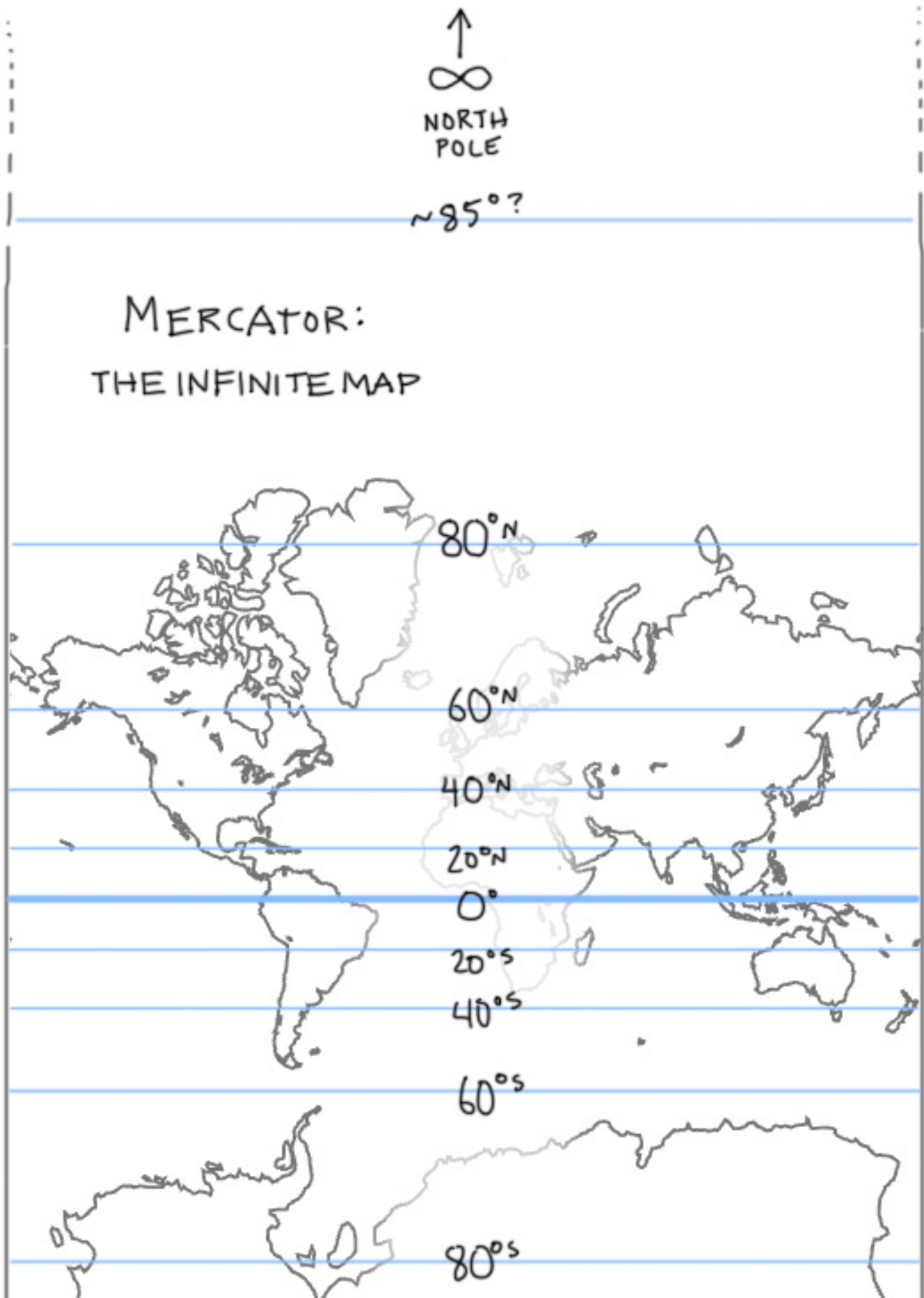
Rhonda is Mapzen's technical publications manager and writes about maps and developer tools.

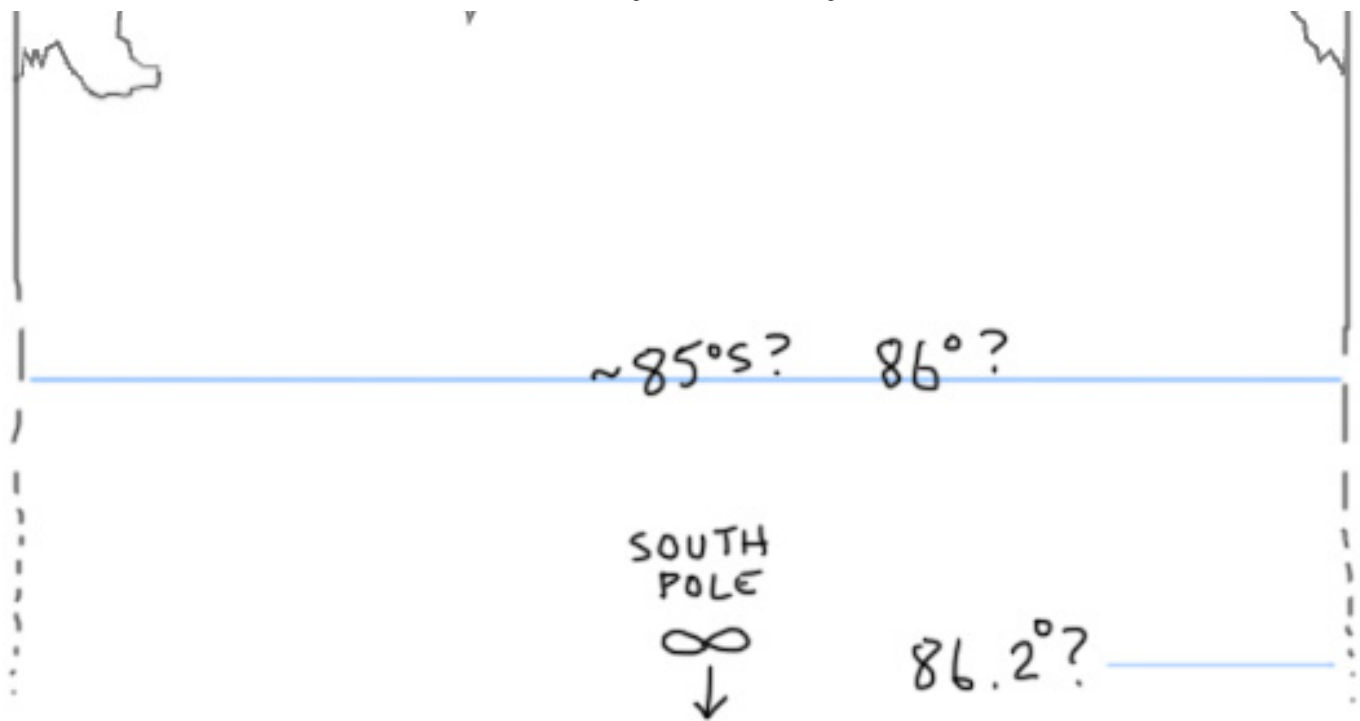
© 2017 Mapzen

Escape from Mercator

tangram (</tag/tangram>) **demo** (</tag/demo>)

Like most web mapping services, our **Tangram** (<http://github.com/tangrams/tangram>) library draws maps in the “**Web Mercator** (https://en.wikipedia.org/wiki/Web_Mercator)” projection. This projection has its benefits, but it’s certainly quirky. Mercator projections are well-known for their distortion at high latitudes – this is because they place the north and south poles at positive and negative infinity, which means a Mercator map of the whole world would be infinitely tall.





To save space (and because there aren't that many points of interest near the poles), all Mercator maps have an arbitrary cutoff point, typically somewhere around 80 degrees. The Web Mercator specifically picks cutoff points so that the whole map makes a perfect square, which has one major benefit: programmers love it.

A square map can be evenly subdivided into increasingly small "tiles", which are easy to serve, request, and use online. This is particularly useful at high zooms (like zoom 16+), when the distortion isn't as obvious. And as most web maps are concerned with smallish, non-polar areas, Web Mercator has been the default projection for the web since Google Maps first popularized it in 2005.

Though it is ubiquitous online (and historically useful to navigators), Mercator doesn't get much love from the modern cartographer. And in general, Mercators are unsuited for cases when you want to compare the size or shape of anything that isn't near the equator. So while Web Mercator is useful, we've been using Tangram to explore other options.

Tangram draws maps in real-time in your web browser, using a hotline to your graphics card called **OpenGL** (<https://en.wikipedia.org/wiki/OpenGL>). Small programs called "shaders" allow the position and coloring of anything onscreen to be modified instantly, according to your own design.

...Did I say "position"? Why yes, yes I did.

(This map is interactive! Open full screen ↗ (<http://meetar.github.io/projection-tests/?wavy.yaml#2/0/0>))

This is the same **vector data** (<https://github.com/mapzen/vector-datasource/>) as in most of our other demos, but the position of each data point is being modified with a single line of shader code in a Tangram scene file, which looks like this:

```
position.y += sin(u_time + position.x/EARTH_RADIUS * 2.) * EARTH_RADIUS / 2.;
```

This is a mathematical function which describes how to move points on a map from one place to another, which also happens to be a very general definition of a map projection.

Most projections are just a series of trig functions which describe how to warp one kind of mapping (generally a sphere) to another (generally a plane). So once you know how to translate those functions to OpenGL, a new Mercator-free world becomes possible using the same method as the wavy map, including everybody's favorite US-centric equal-area conic projection, the **Albers** (https://en.wikipedia.org/wiki/Albers_projection):

(This map is interactive too! **Open full screen** ↗ (<http://meetar.github.io/albers/#5/39.538/-97.603>))

This version of the Albers adjusts depending on your position on the globe. The math is more complicated than the wavy map, but it's still well within the capabilities of older smartphones. Try dragging the map around – you'll see as you navigate that the tiles are still loading as though they were in a standard Web Mercator map, so additional tile-loading code would be needed to fill all the gaps, but it works!

Like most projections, the Albers projects an image of a three-dimensional sphere onto a two-dimensional plane. However, Tangram has another dimension at its disposal:

*(All these maps are interactive! Don't stare at this one too long. **Open full screen** ↗
(<http://meetar.github.io/projection-tests/?globe-warp.yaml#1/0/0>)*

The above map restores our 2D Web Mercator data to its rightful place in three dimensions, in an actual 3D scene, courtesy of the 3D capabilities of OpenGL – and as Tangram is a Leaflet plugin, you can still pan and zoom this map like any other web map. But why stop there?

From the description of **BERG's Here & There project**
(<http://berglondon.com/products/hat/>):

...the ability to be in a city and to see through it is a superpower, and it's how maps should work.

We tend to agree. As an example, the map below is an homage to the **Here & There maps** (<http://berglondon.com/products/hat/>), with a little **Inception** (<https://en.wikipedia.org/wiki/Inception>) thrown in:

(*Open full screen ↗* (<https://meetar.github.io/bendy-map/#14/40.7238/-73.9881>))

This bendy map is a mix of two views of the same data: the top part of the map is a standard top-down web map view (plus 3D buildings), and the bottom part is a tilted view of the same scene. A shader then cross-fades between the two views, based on distance from the camera, while you pan around the map.

And this next one rolls those same vector tiles into a tiny planet, by wrapping the tiles into a sphere of constant size and trimming away whatever is left over. Drag it to roll New York up into a beautiful **katamari** (https://en.wikipedia.org/wiki/Katamari_Damacy):

(*Open full screen* ↗ (<http://meetar.github.io/projection-tests/?katamari.yaml#15/40.7477/-73.9866>))

Caveat Everybody

Wet-blanket time: these maps are *~highly~* experimental, and not production-ready. The global projections get shaky past zoom 15 because of floating-point precision limits, we don't have tile-fetching worked out for them yet, and they don't work with our current labels system – but we're working to address all of these issues, and as always, **we encourage contributions** (<https://github.com/tangrams/tangram/blob/master/CONTRIBUTING.md>)! In the meantime, we think these experiments are useful for demonstrating the ways open data viewed with OpenGL can expand our cartographic horizons.

· 26 October 2015 ·



Peter Richardson

Peter vexes vertices, flusters fragments, and pesters pixels on Mapzen's graphics team.

© 2017 Mapzen

Viva La Revolution de Carta

This month's witching hour takes me to Mexico City not for my awesome trick or treat costume (photos coming soon!!) but for the **Open Government Partnership Summit** (<http://ogpsummit2015.sched.org/>) from October 27-29th.



The Summit brings reformers from 66 countries to talk strategies in making their governments more open, accountable, and responsive. Since open is part of my jam... I'll be there! With many from the open geo community we hope to build momentum for further cooperation, exchange and action when it comes to mapping. If you are also going to be there, come find me! When not covered in OpenStreetMap stickers I'll be speaking at the following sessions:

- **Wednesday, October 28 * 12:30pm-1:30pm:**

I'll assist at the **OpenStreetMap Mapathon**

(<http://ogpsummit2015.sched.org/event/cdfe292b41e2464e64fd058ddeb1bbbe#.Vi4iXRCrTsk>) with friends from the open geo community.

- **Wednesday, October 28 * 6:00pm-7:00pm:**

I'll be geo speed geeking (not speed dating) at **Geo Open Data 4 Official Stats**

(<http://ogpsummit2015.sched.org/event/065c154f1efb1b0113474116d9056be1#.Vi4oSRCrTsk>)

- **Thursday, October 29 * 10:00am - 11:00am:**

Panel on the Global Impacts of Open (geo) Data

If you're not at the Summit but want to keep up with all the mapping events keep an eye on **#ogpmapping** (<https://twitter.com/search?src=typd&q=ogpmapping>) and **#mappingrevolution** (<https://twitter.com/search?q=mappingrevolution&src=typd>). Because we expect no less than a revolution when it comes to open mapping.

pumpkin via Attleboro Farmers' Market** (<http://attleborofarmersmarket.com/2013/01/03/animated-gif-pumpkin-world/>), **pumpkin by the Polis family

· 26 October 2015 ·



Alyssa Wright

Alyssa Wright does community where the phone and meeting are her superpowers of choice.

A Frighteningly Open Halloween Map

Many Halloween maps are scary, but ours is FRIGHTENINGLY OPEN – **you too can contribute** (<https://github.com/mapzen/halloween-map>)!



(This map is interactive! Open full screen ↗ (<https://mapzen.com/maps/halloween/>))

We took one of our base scenes called **grain** (<https://github.com/tangrams/tangram-sandbox/blob/gh-pages/styles/grain-area.yaml>) and added halloween-styled color to the hatching on land use areas, and then took the logic from our **tron** (<https://github.com/tangrams/tangram-sandbox/blob/gh-pages/styles/tron.yaml>) style to make the rivers run red with blood. There's a javascript function **inside the yaml scene file** (<https://github.com/mapzen/halloween-map/blob/gh-pages/halloween.yaml#L415>) that does spooky substitution on the street names. POI icons are by multiple authors from the **Noun Project** (<https://thenounproject.com/>).

Is there something you want to add? **Make a pull request** (<https://github.com/mapzen/halloween-map>) on the public Github repo! We'd love to see what you come up with.

· 30 October 2015 ·



Mapzen

Opening maps with open source and open data.

© 2017 Mapzen

An Open Directory of Transit Data Sources: Transitland's Feed Registry

transitland (/tag/transitland)

Public transit works best without barriers, when it's simple for riders to transfer from one route to another, from one operator to another, and from one ticket and fare system to another. Similarly, transit data works best without barriers, when it's simple for routing engines, visualizations, and analyses to mix data from multiple sources and transit operators.

Transitland (<https://transit.land>) is an experiment sponsored by Mapzen to aggregate transit data from around the world and make it seamless to use together—both technically and legally. We've been developing **the Onestop ID scheme** (<https://github.com/transitland/onestop-id-scheme>) to enable users of all experience levels to work with stops and routes across multiple operators and datasets. Anyone can use a Onestop ID to look up transit data through the **Datastore API** (<https://transit.land/how-it-works/#slide-3>) or the **Playground data explorer interface** (<https://transit.land/playground>).

We're now pleased to offer another view into Transitland's collection of data, the **Feed Registry** (<https://transit.land/feed-registry>). We've created the Feed Registry to help the users of open transit data answer two questions:

1. ***What transit operators offer open feeds?***
2. ***What can I legally do with each feed?***



HOME

HOW IT WORKS

FEED REGISTRY

PLAYGROUND























AN OPEN PROJECT

PARTICIPATE

FEED REGISTRY

The Feed Registry lists all the transit operators and feeds included in Transitland. See what each feed's license will allow you to do with it and click through to get more details about the data included in the feed.

1 country
9 agencies

| Agency Name | Region | State | Country | License | License Details |
|--|------------------------|------------|---------------|---|---|
| Alameda-Contra Costa Transit District | San Francisco Bay Area | California | United States | Creative Commons Attribution 3.0 Unported License |    |
| Bay Area Rapid Transit | San Francisco Bay Area | California | United States |  |    |
| Caltrain | San Francisco Bay Area | California | United States |  |    |
| MTA New York City Transit | New York City | New York | United States |  |    |
| New York City Department of Transportation | New York City | New York | United States |  |    |
| Port Authority Trans-Hudson | New York City | New York | United States |  |    |

(<https://transit.land/feed-registry>)

What transit agencies offer open feeds?

The Feed Registry provides an overview of all the operators represented in Transitland. For every operator, we include a list of the feeds we aggregate from each. Most transit operators only offer one feed, but some, like **MTA New York City Transit** (<https://transit.land/feed-registry/operators/o-dr5r-nyct>), are split across many feeds.

Want to use the Feed Registry's data in your own application? It's all available through the Datastore API under the `/api/v1/operators` (<https://transit.land/api/v1/operators>) and `/api/v1/feeds` (<https://transit.land/api/v1/feeds>) endpoints. We're assembling and sharing

this data under **Creative Commons CC0**

(<https://creativecommons.org/publicdomain/zero/1.0/legalcode>) (CC0), so you're free to use this catalog of operators and feeds however you see fit in your own applications and services.

For now, we only list transit operators in San Francisco and New York City. Soon, we'll be expanding our coverage worldwide, and we'll be opening the Datastore API to outside contributors to add additional operators and feeds. Do you work for a transit operator that is looking to share its data more widely? Or are you a transit data enthusiast interested in contributing to Transitland? Please **write us** (<mailto:transitland@mapzen.com>), and we'll add you to our list of beta testers.

What can I legally do with each feed?

Operators release their transit data under a variety of licenses. To work with one feed at a time isn't very difficult; it's just a matter of reading the license and any other terms provided by the operator. But to work with more than one feed at a time becomes a challenge. Not only are there many licenses to read, but each allows slightly different permissions and imposes slightly different restrictions.

With the **Feed Registry** (<https://transit.land/feed-registry>), we're hoping to clarify this puzzle of licenses. For each operator, we summarize three aspects about their license:

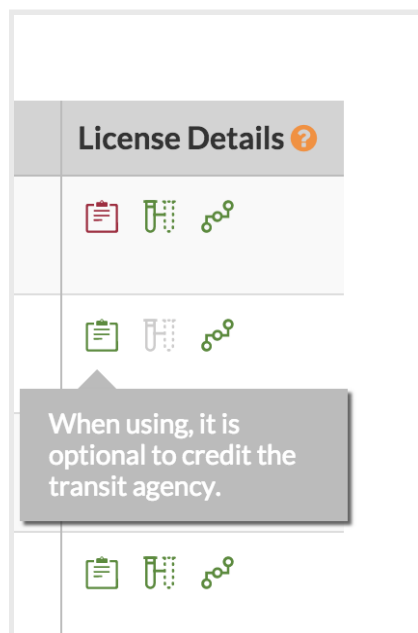
can you use this feed without attribution? Or do you need to explicitly mention in your application, visualization, or analysis that you're making use of data from this operator?

can you create derived products from this feed? For example, can you mix the feed with other sources and share the mixed file with others?

can you take this feed and redistribute it to others, in a different context or embedded within a different application?

Each license is summarized with three icons that are color-coded for quick reference. Hold your cursor over an icon in the Feed Registry to get more information about what that license permits, or click the question mark at the top of the "License Details" column for more information. Even with just these feeds from the San Francisco and New York regions, it's possible to see how each license is "open" in a slightly different way.

Please note the Feed Registry is provided for informational purposes only and does not constitute legal advice. We make a good faith effort to ensure accuracy, but cannot guarantee the accuracy or reliability of the information provided in the Feed Registry. You are advised to click through to each feed's license, review it, and consult with a lawyer if you need proper legal advice on using and consuming the data referenced in the Feed Registry.



(<https://transit.land/feed-registry>)

Your edits and corrections are welcome. Please **contact us** (<mailto:transitland@mapzen.com>).

We've already mentioned how we're planning to work together with the producers and consumers of transit data to grow the Feed Registry's coverage. We're also working together with the producers and consumers of transit to increase the number of rows in the Feed Registry that feature green license icons—in other words, that transit data is as open as possible.

Do you work at a transit agency that is hoping to make its data more open and useful to civic-minded software developers and entrepreneurial companies? We're glad to help you make all of those icons "go green." Our lawyer has drafted a model license that allows the freedoms that developers want, while also including the protections that public transit agencies need. Please visit the **Transitland website for more information about the model license** (<https://transit.land/an-open-project#for-data-providers>).

Are you a consumer of transit data hoping to improve the accessibility of a feed? Consider writing to the feed's owner, explaining the value of open transit data, pointing them to the **Transitland website for more information about the model license** (<https://transit.land/an-open-project#for-data-providers>), and **letting us know** (<mailto:transitland@mapzen.com>) what kind of response you receive. We just ask that you approach all transit agencies with respect. It's only by working together that we'll be able to join together transit data from around the world and make it truly accessible.

· 04 November 2015 ·

**Drew Dara-Abrams**

Drew leads Mapzen Mobility products (and aspires to being a flâneur).

**Meghan Hade**

Meghan is a software engineer with a background in urban planning on Mapzen's Mobility team. She works in javascript and plays in calligraphy, block printing, and finding ways to stash n+1 bikes in a San Francisco apartment.

© 2017 Mapzen

Searching for Geoweb in Brooklyn

search (/tag/search)



The **11th annual Geoweb Summit comes to Brooklyn** (<http://geowebsummit.com/>) – the heart of so much geo, web, and coffee advancements — on November 12th and Mapzen will be there!! Diana Shkolnikov, our director of engineering for **search**

(<https://mapzen.com/projects/search>), will start off the fun with a panel of open data and tools with speakers from AirBnB and CartoDB. Learn more about how Mapzen Search works with open data communities and open source strategies to make the only totally open, locally-grown, cage-free geocoder. It's the Mapping with Open Data and Tools panel at 4pm. And feel free to bring your own coffee advancements to the fun.

image via **Flickr user Yusuf C, CC 2.0**

(<https://www.flickr.com/photos/90326842@N00/4740417370>)

· 11 November 2015 ·



Alyssa Wright

Alyssa Wright does community where the phone and meeting are her superpowers of choice.

© 2017 Mapzen

Find your community: an exploration of our geocoding plugin

search (</tag/search>)

If you've ever built a map-based application, chances are you've put a location search box somewhere on it before. And if you're using our **Mapzen Search** (</projects/search>) service, you might not even have to build it yourself!

When we launched Mapzen Search **six weeks ago** (</blog/this-is-it-mapzen-search-is-now-live/>), we knew that there was going to be a number of our users who shared a common need for that search box, so we built a **Leaflet geocoder plugin** (<https://github.com/mapzen/leaflet-geocoder>) that would get your map working with Mapzen Search very quickly, even if we've had to choose some defaults for you.

Of course, we can't predict or meet everyone's needs at once. So the plugin is designed so you can customize it beyond the default experience. For instance, you could even direct to another service if you wanted to host your own **Pelias** (<https://github.com/pelias/api>)-based geocoding service.

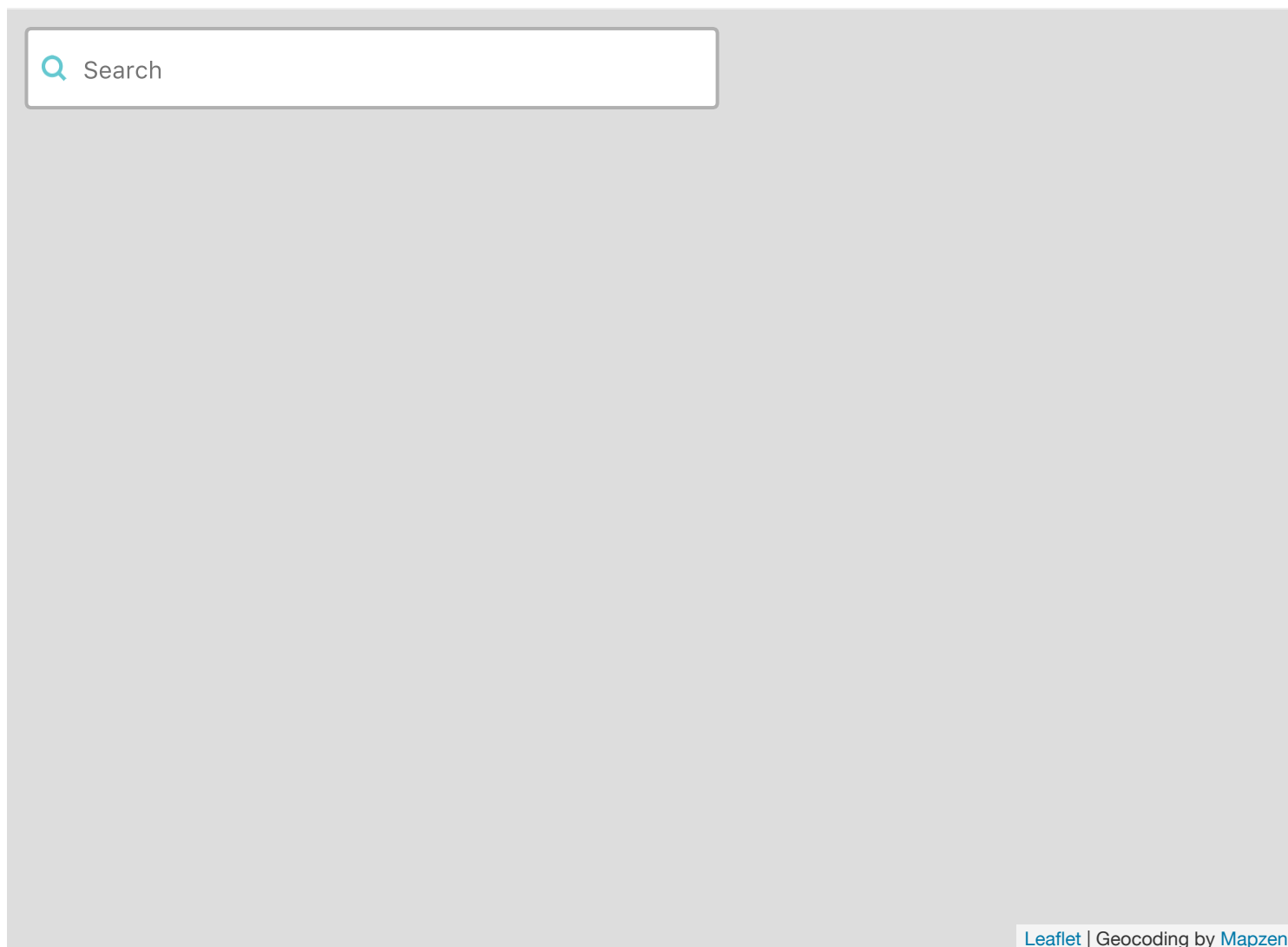
And, as a Leaflet plugin, it should provide an abstraction layer over Mapzen Search and Pelias, so if you are already experienced in Leaflet you should not need to learn an entirely different syntax (until you really need to, of course).

So how did we do?

I decided to put our own plugin to the test.

Welcome to the neighbourhood

First, though, let's get you situated. A basic use of the plugin will drop a search box in the upper-left corner, just like this:



(This map is interactive! Open full screen ↗ (<https://mapzen.github.io/leaflet-geocoder/>))

It's not really flashy, but it's not meant to be. It blends right into Leaflet and it does its job.

Dropping it into your project should be straightforward, as well. If you've already written some of the code to set up a Leaflet map, there are only a few important lines to add to get started. In an HTML file, load the plugin script and stylesheet somewhere after loading Leaflet:

```
<!-- in HTML -->
<link rel="stylesheet" href="leaflet-geocoder-mapzen.css">
<script src="leaflet-geocoder-mapzen.js"></script>
```

Inside of your JavaScript, instantiate your Leaflet map, and then add the geocoder like this.

```
// In JavaScript
// Get an API key from https://mapzen.com/developers/
L.control.geocoder('<your-api-key>').addTo(map);
```

And if you need more help getting started, we've got your back! We've created an in-depth, step-by-step **tutorial for the Leaflet geocoder plugin** (<https://mapzen.com/documentation/search/add-search-to-a-map/>) which takes less than 30 minutes to work through. (This was the topic of a **Maptime Oakland** (<http://www.meetup.com/Maptime-SF/events/226540295/>) event in November 2015.)

Remodeling the homestead

Maybe the basic plugin just isn't enough for you. After all, your project might have a different look and feel, and you want to make that plugin fit your needs, not the other way around.

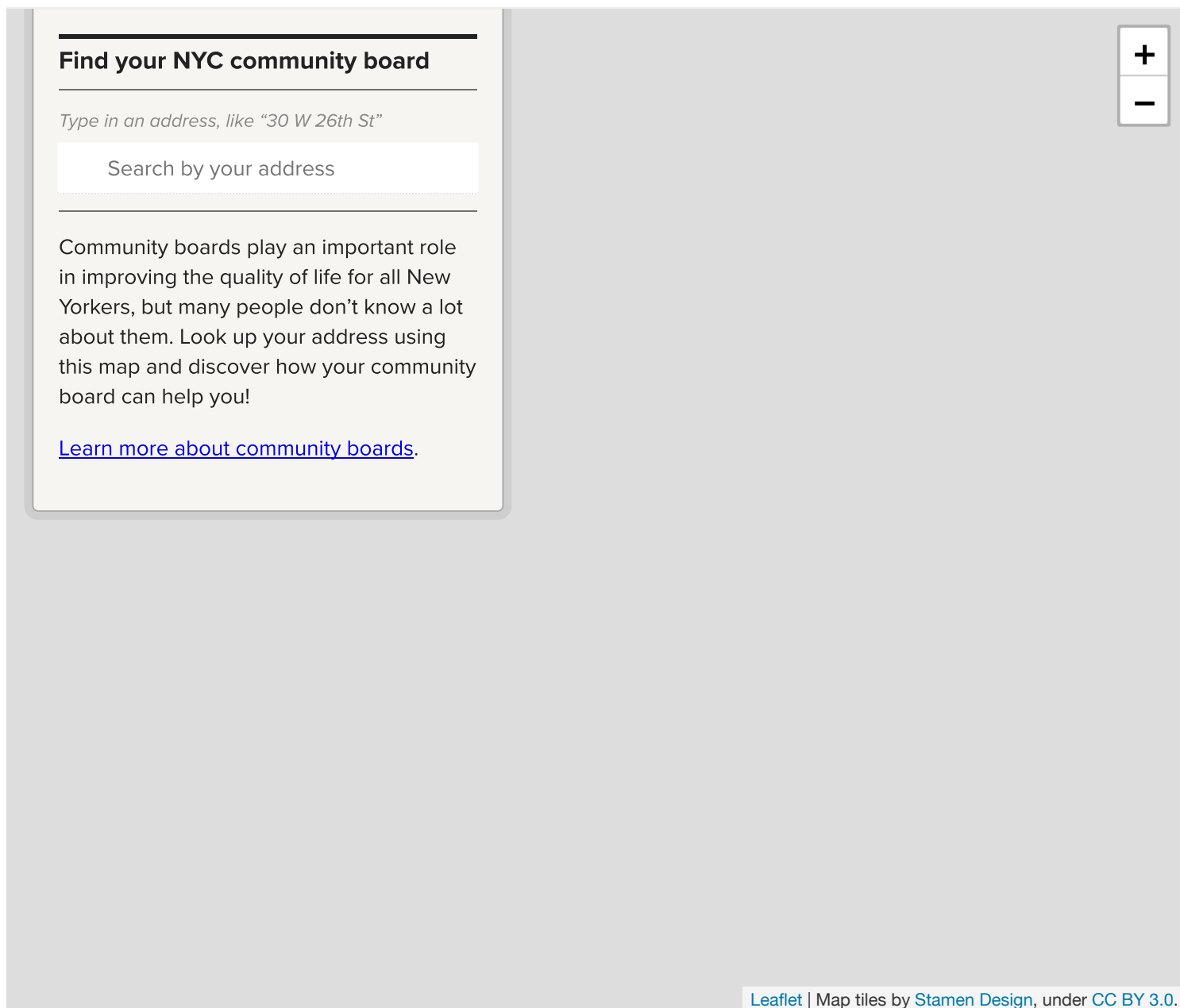
To understand what some of those needs are, I needed to approach it from another perspective, as a developer with other goals in mind. I decided to start with a small project, the kind of "micro-service" a civic technology-minded person like myself might tackle as a side project or at a series of **hack nights** (<http://www.codeforamerica.org/blog/2013/07/24/brigade-101-how-to-hack-night/>).

I was recently inspired to learn more about my own neighborhood's community board, after attending the **Municipal Arts Society (MAS) Summit** (<http://www.summit.mas.org>) in New York City, where I now live. I saw how these community boards play a large part in representing neighborhoods and their residents when there are significant projects or policies that impact these communities.

But the city's own **Find Your Community Board**

(<http://www.nyc.gov/html/cau/html/cb/cb.shtml>) page, while informative, didn't make it as easy as they could have. I was reminded of **ANC Finder** (<http://ancfinder.org/>), a community-built project in Washington, D.C. (where I used to live, also), where you could enter an address and find out which ANC, or Advisory Neighborhood Commission, represented you. These ANCs play a similar role in local politics for DC residents.

So I decided to build a small prototype project that would use the geocoder plugin so New York City residents like myself could easily find out about the community boards that represented us. A few days later, here's the result.



(This map is interactive! Open full screen ↗ (<http://louh.github.io/nyc-community-boards/>))

Since this was just a prototype, I found myself cutting out a lot of features that I otherwise would have included if this was more of a long-term project. Of course, if you'd like to fork this project or contribute to it, the code is totally open and **hosted on GitHub** (<https://github.com/louh/nyc-community-boards>).

But now, I'd like to talk a little bit about what I've learned incorporating the plugin on a small project.

Getting to know the neighbours

Fitting the plugin into a different layout was, thankfully, not too difficult. The question I wanted to answer was this: *Could I put the search box inside of another box?* It did involve manually positioning the search box over another element that existed outside of the map, so it's not effortless, but it's doable.

Also, since I could only match up New York City addresses with community board boundaries within the city, I had to structure my search query to be bounded in order to be useful. Since the plugin translated Leaflet syntax into an appropriate query for Mapzen Search behind the scenes, I couldn't just ask the plugin to filter results by a locality. So I could only pass in a rectangular bounding box containing New York City's boundaries, but that meant some results from New Jersey could show up in the results. I decided to capture these results by showing the marker anyway but providing a helpful error message.

Another problem showed up in the autocomplete feature. The Mapzen Search API `/autocomplete` endpoint allows you to boost results around a certain focus point. However, the plugin had never included support for this functionality. For this project, auto-suggested results were coming from all over the world, which gave very confusing feedback for the user. So, for now, I simply turned autocomplete off. This is something we'll look into adding to the plugin as well.

Lastly, I've found that when users perform actions with the plugin, a developer needs to have some hooks so that they can piggy-back on top of (or override!) the search box's actions. For instance, I wanted to make sure that even though a marker is placed, the default behavior of showing a popup is suppressed. Another case is when a user clears the search result box. For this project, it removed the marker but not the community board boundary or information, which felt strange.

To overcome these problems, I had to **overwrite the plugin's internal methods** (<https://github.com/louh/nyc-community-boards/blob/gh-pages/src/js/main.js#L154-L202>) to make it do what I wanted. This was doable because I was familiar with the guts of the plugin, but none of this is documented behavior, and it could even change without warning in the future! So it's clear that for these actions, we need to expose and document hooks that developers can use, and make sure that they are covered by tests so that they remain reliable for years to come. Stay tuned for documentation and a blog post on these hooks.

Civic engagement!

Of course, none of these should prevent you from using the Leaflet geocoder plugin in your own projects today if it's the right fit for you. Some people already are! **Map My Story** (<http://www.mapmystory.xyz/>), a project that traces ancestral migration paths that came out

ChiHackNight (<http://chihacknight.org/>), uses the plugin. So does the **Greenpoint-Williamsburg ToxiCity Map**

(http://clhenrick.github.io/greenpoint_williamsburg_toxicity_map/), which shows potentially polluted sites in these neighborhoods, and **what3emojis** (<http://what3emojis.com/map/>), a way to create an address using emoji, and may or may not be a real project (full disclosure: I am an early angel investor of it).

So we hope you find the plugin, and Mapzen Search in general, to be the least painful part of your application development. And if you run into any questions or suggestions for improvement, **please let us know** (<mailto:pelias@mapzen.com>)! We love hearing from you.

And if you're a resident of **Fort Greene** (<http://louhuang.com/nyc-community-boards/?query=Fort%20Greene%2C%20Brooklyn%2C%20NY&lat=40.69371&lng=-73.96708>), maybe I'll see you at a community board meeting sometime!

· 12 November 2015 ·



Lou Huang

Lou is a protagonist of an open world.

© 2017 Mapzen

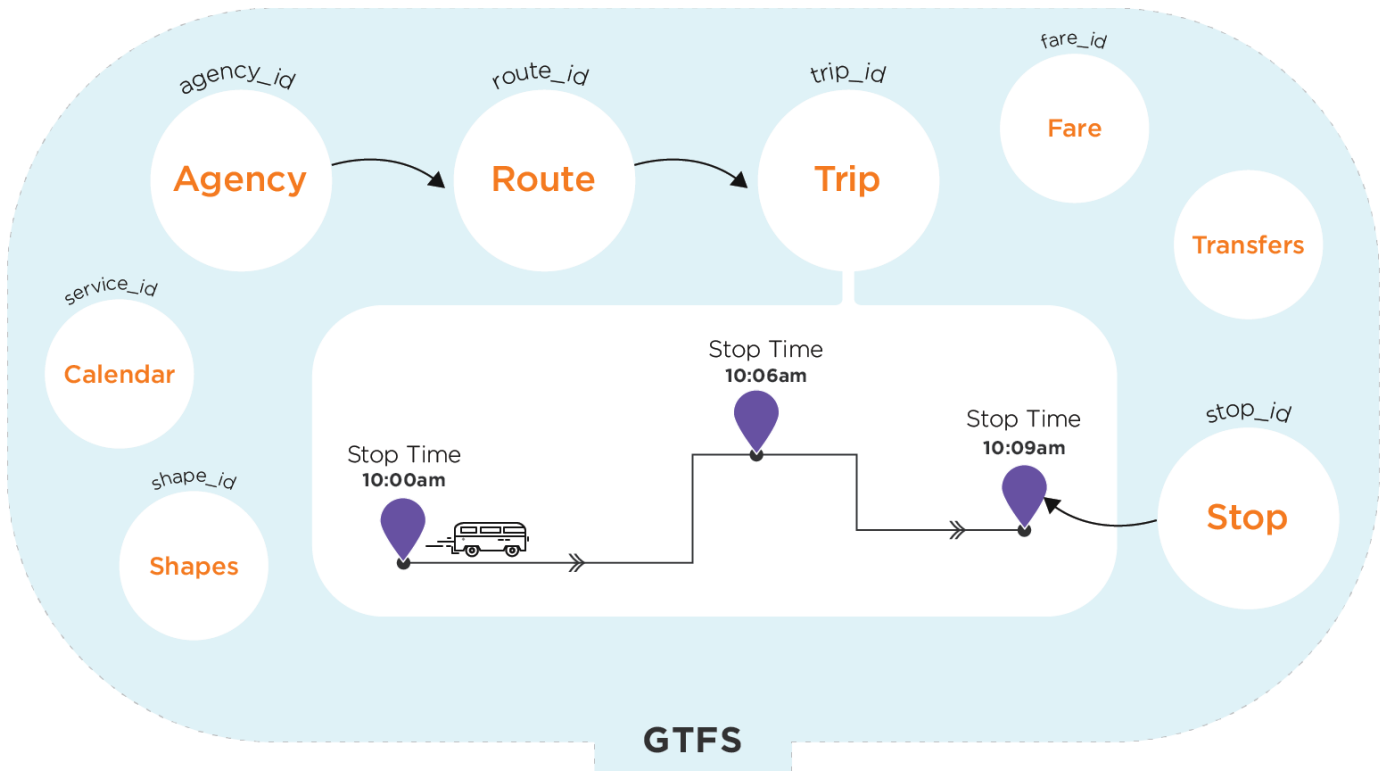
Transit dimensions

transitland ([/tag/transitland](#))

Trains and buses are cool. They free your attention, widening your experience of a city. The best routes travel through space on dedicated rights of way, avoiding the tedium and frustration of traffic. Transit is also dynamic, a ballet of thousands of vehicles moving to complex schedules, and a good transit map needs a time dimension to reveal hidden networks of speed, frequency, and service. At Transitland, we have built a transit schedule API, a new resource to power routing applications, transit visualizations, and wonderful tools we hope will be realized with community vision.

Ten million tiny movements

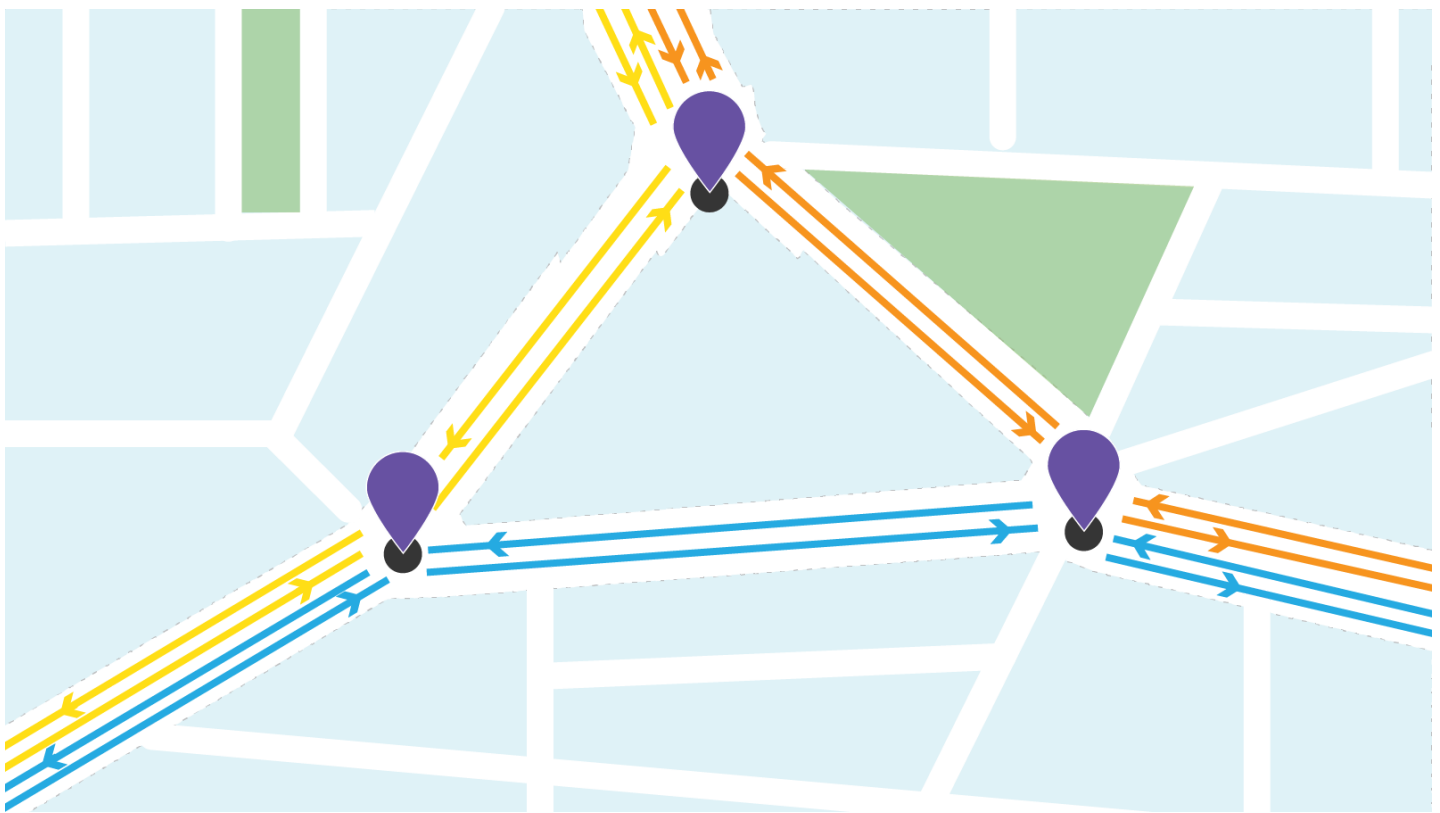
Open transit data comes from a variety of sources, most commonly using the **General Transit Feed Specification (GTFS)** (<https://developers.google.com/transit/gtfs/reference?hl=en>) developed by **TriMet** (<http://trimet.org/>) and Google in 2005. This collaboration created a versatile specification, balanced between the needs of transit agencies and data consumers. GTFS has added features and extensions over the years, but the core specification is remarkably resilient, and has been widely adopted by hundreds of transit agencies across the world.



GTFS describes a transit system as tables of agencies, routes, stops, and trips that individual transit vehicles make according to a set schedule. Each trip begins at a stop, then visits another stop, and so on, until the trip ends at a final stop. Each stop on a trip, or stop_time, includes an arrival time, a departure time, and accessibility details. This point-to-point representation is intuitive, and easy to describe as a series of rows in a CSV file or relational database.

Transitland schedule API

However, there are other useful ways to think about a transit schedule. Fundamentally, transit is about moving between stops as efficiently as possible, and any two stops may have many possible connections — through different routes, at different times of day, using weekends and holidays schedules, etc. The Transitland schedule API transforms GTFS schedule data into a set of connections between stops, representing the transit network as a large directed graph.



In this representation, each possible move between two stops has a unique edge, called a `ScheduleStopPair`. **Each edge includes** (https://github.com/transitland/transitland-datastore/blob/master/doc/schedule_api.md) an origin stop, a destination stop, a route, an operator, and arrival and departure times. Each edge also includes a service calendar, describing which days a trip is possible. Accessibility information for wheelchair and bicycle riders is included, if available. Some of this data is normally split across multiple GTFS tables, but is here denormalized for simpler access: each edge contains enough information to get from one stop to another, to another, and finally to your destination.

Querying the schedule API

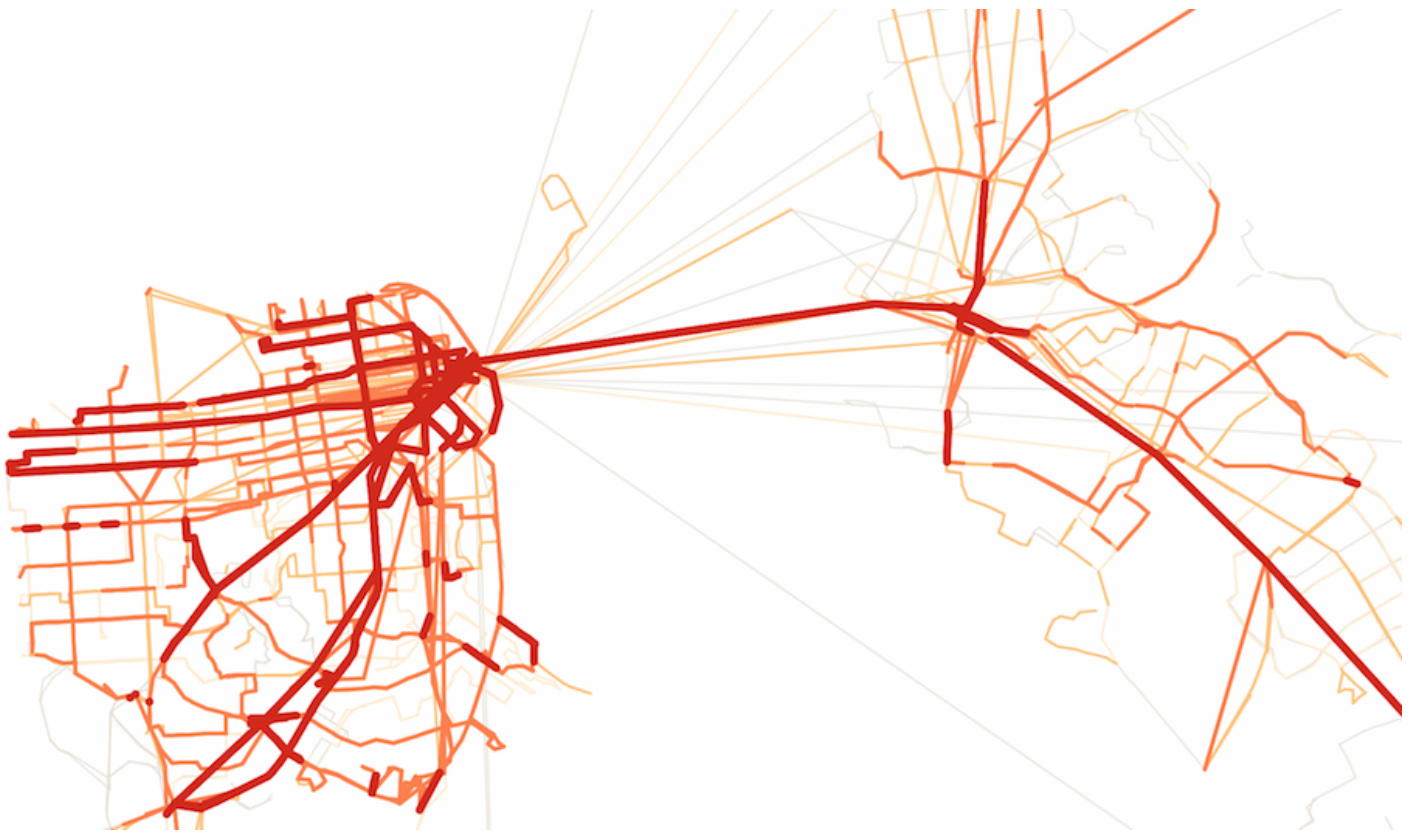
The **schedule API endpoint** (http://transit.land/api/v1/schedule_stop_pairs) allows you to search the graph in several useful ways. For instance, you can request only data within a particular geographic region. Or, for a given day, or time of day, or a specified time period. Fine-grained access to schedule data frees you from having to download, parse, and manage many different GTFS schedules (each of which may have millions of rows) and instead focus on the data most relevant to your application. A few example queries are provided below; please visit the **schedule API documentation** (https://github.com/transitland/transitland-datastore/blob/master/doc/schedule_api.md) for additional details.

| Query parameter | Description | Example |
|--------------------------|--|--|
| origin_onestop_id | Origin Stop | from Embarcadero BART
(http://transit.land/api/v1/schedule_stop_pairs?origin_onestop_id=s-9q8znb12j1-embarcadero) |
| destination_onestop_id | Destination Stop | to Montgomery St. BART
(http://transit.land/api/v1/schedule_stop_pairs?destination_onestop_id=s-9q8yyxq427-montgomeryst) |
| route_onestop_id | Route | on Muni N
(http://transit.land/api/v1/schedule_stop_pairs?route_onestop_id=r-9q8y-n) |
| operator_onestop_id | Operator | on BART
(http://transit.land/api/v1/schedule_stop_pairs?operator_onestop_id=o-9q9-bart) |
| service_date | Service operates on a date | valid on 2015-10-26
(http://transit.land/api/v1/schedule_stop_pairs?date=2015-10-26) |
| service_from_date | Service operates on a date, or in the future | valid on and after 2015-10-26
(http://transit.land/api/v1/schedule_stop_pairs?service_from_date=2015-10-26) |
| origin_departure_between | Origin departure time between two times | departing between 07:00 - 09:00
(http://transit.land/api/v1/schedule_stop_pairs?origin_departure_between=07:00:00,09:00:00) |
| trip | Trip identifier | on trip '03SFO11SUN'
(http://transit.land/api/v1/schedule_stop_pairs?trip=03SFO11SUN) |
| bbox | Origin Stop within bounding box | in the Bay Area
(http://transit.land/api/v1/schedule_stop_pairs?bbox=-122.554,37.668,-122.085,37.912) |

Frequency is freedom: an example

Jarrett Walker (<http://www.humantransit.org/frequent-networks/>) and many others advocate that frequent service is the foundation of a robust transit network, and critical for rider trust. Yet, frequency information is often **missing or obscured** (http://www.actransit.org/pdf/maps/version_29/city_map.pdf) on official transit maps, or

manually drawn by enthusiasts (<http://calurbanist.com/east-bay-frequent-transit/>) to **fill the gap** (<https://www.sfmta.com/projects-planning/projects/new-muni-map>). A GTFS schedule contains all the data necessary to create a frequent service map, but regional transit service is often split among many agencies and multiple GTFS feeds.



A transit intensity map of San Francisco and Oakland, typical Monday 6-9am peak commute hours. Stronger color and line width represents more frequent service between two stops.

The Transitland schedule API can bridge these gaps, providing the data necessary to analyze complex transit patterns across a large region. I created a **Python script** (<https://gist.github.com/irees/272e5dc57614cab595a0>) to generate the above visualization and serve as an example of the schedule API in action. A combination of query parameters (**date, time period, and bounding box** (http://transit.land/api/v1/schedule_stop_pairs?date=2015-10-26&origin_departure_between=07%3A00%3A00%2C09%3A00%3A00&bbox=-122.554%2C37.668%2C-122.085%2C37.912)) filters the schedule data, and the number of trips between each two stops is counted. The coordinates for each stop are then used to generate a **GeoJSON map** (<https://gist.github.com/irees/f9a4d9d27e202309e9de>), with line width and color showing the number of trips per hour.

Explore with us

Hop on the bus and explore the possibilities of a schedule API with us. It's more fun together! Schedule data for the SF Bay Area feeds and NYC subways is now available for all through the Transitland Datastore API (with more coming soon!) The **schedule API documentation** (https://github.com/transitland/transitland-datastore/blob/master/doc/schedule_api.md) and **example script** (<https://gist.github.com/irees/272e5dc57614cab595a0>) provide a few departure points, and we'd love to hear where you'd like to go next.

· 13 November 2015 ·



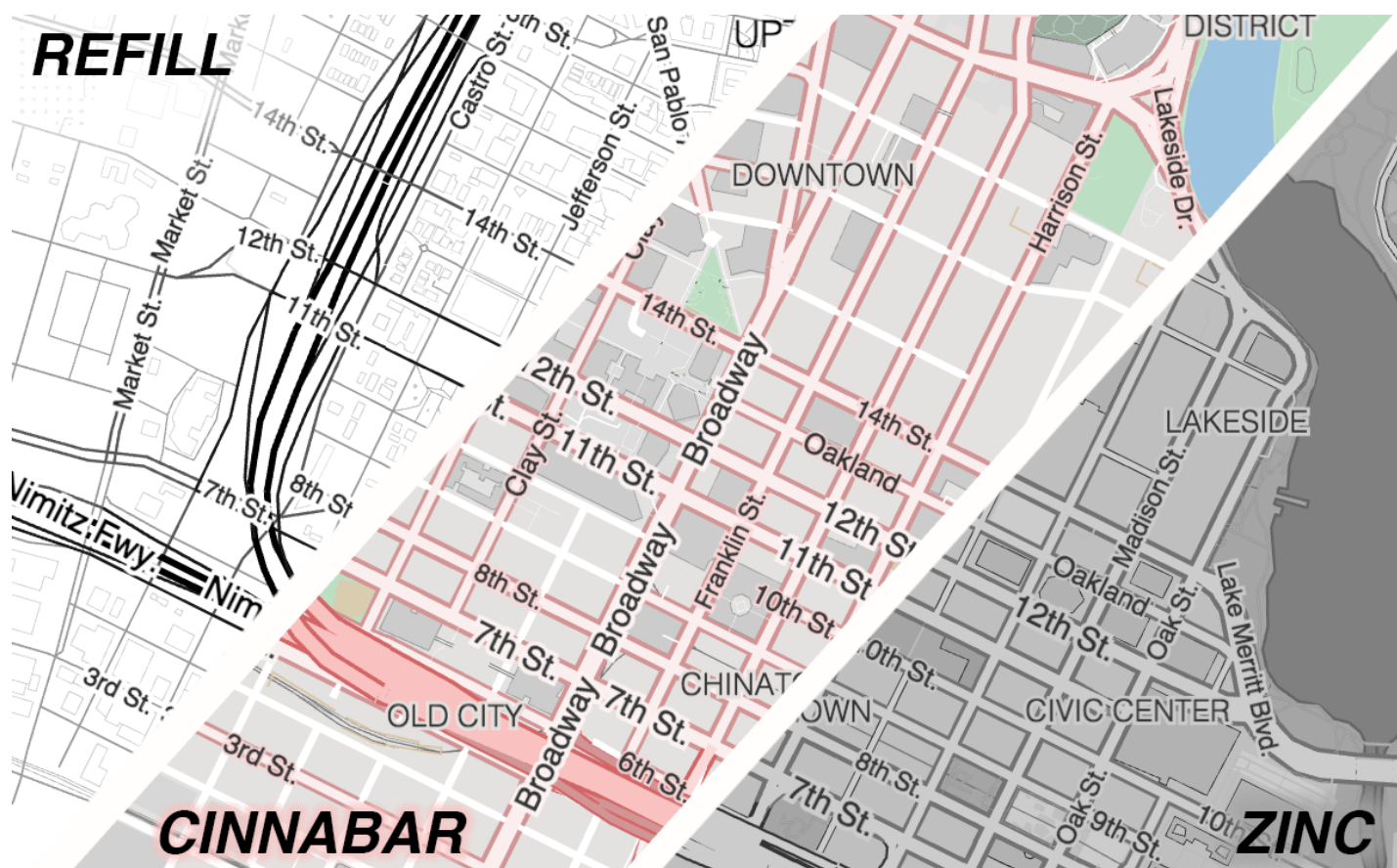
Ian Rees

Ian spends too much time thinking about cities, land use, and open transit data on the Mapzen Mobility team.

© 2017 Mapzen

A family of styles with many flavours: Introducing Refill, Cinnabar, and Zinc styles for Tangram

`tangram` (`/tag/tangram`) `vector-tiles` (`/tag/vector-tiles`) `cartography`
(`/tag/cartography`)



At Mapzen we're interested in radically improving access to open data. Having more open geo data is great and is something we're working hard on with projects like **Who's On First** (<https://whosonfirst.mapzen.com>), **Metro Extracts** (<https://mapzen.com/data/metro-extracts/>) and contributions to **OpenStreetMap** (<http://www.openstreetmap.org>). But it's just as important to us that **mappers should have easy access to open data so you can focus on creating great products and experiences, and not get stuck wrangling data.**

Please use and adapt these open source scene files in your own projects. **We're excited to see what you build!** Let us know @mapzen (<http://twitter.com/mapzen>) :)

A family of styles ...

The following styles are all built on the same skeleton of map features (curated by zoom), but each is skinned for a different cartographic goal. Even though they look different, via the power of boolean visibility and color variables they are only a few dozen lines different. **You don't need to be a Tangram styling pro to remix these styles.** We'll blog more about that soon.

Refill



The first style is called **Refill** and has been previewed in some Mapzen projects already like the Who's On First **Spelunker** (<https://whosonfirst.mapzen.com/spelunker/>). Refill provides a high contrast, black & white basemap useful for data visualization.

Refill is a continuation of the **Toner** (http://content.stamen.com/dotspotting_toner_cartography_available_for_download) style Geraldine started at **Stamen** (<http://stamen.com>). With the GL capabilities of Tangram, detailed lines, patterns and building extrusions are further explored and elaborated upon from the previous Toner base-style. Toner was a style created at Stamen as part of their great **CityTracking** (<http://citytracking.org>) project.

Cinnabar



The second style is called **Cinnabar** and is a **High Road** (<https://github.com/migurski/HighRoad>) influenced evolution of the **Traditional** (<http://tangrams.github.io/tangram/#mapzen,40.70531887544228,-74.0097749233246,16>) style Stamen created for Mapzen's **Open** (<https://mapzen.com/blog/we-made-an-app>) Android app last year. This classic style should be your go-to for general mapping applications.

Zinc



Rounding out the set is **Zinc**. Like Refill, this style is designed for data visualization. Mapzen's Zinc style is comparable to Esri's **light gray Canvas** (<http://blogs.esri.com/esri/arcgis/2011/09/29/esri-canvas-maps-part-i-author-beautiful-web-maps-with-our-new-artisan-basemap-sandwich/>) style, Mapbox's **Light** (<https://www.mapbox.com/blog/custom-styles-mapbox-streets/>) style, and CartoDB's **Positron** (http://content.stamen.com/new_maps_for_cartodb) style, created by Stamen. If you like any of those, Mapzen's Zinc basemap style is for you.

... with many flavours

Mapzen offers each style in three flavours, giving you 9 different styling options. Yummy, time to make some (vector) **map sandwiches** (<http://blogs.esri.com/esri/arcgis/2009/07/13/the-map-sandwich/>)! Curious about map sandwiches? CartoDB has a more **recent post** (<http://blog.cartodb.com/let-your-labels-shine/>).

1. **Default** - Some labels for streets, cities, water bodies, and some big parks with name only (no icons). No business labels. Good for data visualization overlays that need to provide

some location context.

2. **More labels** - Fuller set of labels, including high contrast icons highlighting OpenStreetMap business listing data.
3. **No labels** - Just the lines and polygons, please.

The Matrix – live demos and source code

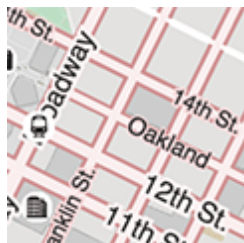
- **Refill**

- **Default:** preview map (<http://tangrams.github.io/refill-style>) (view source on Github (<http://github.com/tangrams/refill-style>))
- **More labels:** preview map (<http://tangrams.github.io/refill-style-more-labels>) (view source on Github (<http://github.com/tangrams/refill-style-more-labels>))
- **No labels:** preview map (<http://tangrams.github.io/refill-style-no-labels>) (view source on Github (<http://github.com/tangrams/refill-style-no-labels>))



- **Cinnabar**

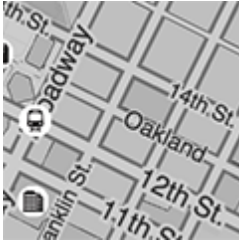
- **Default:** preview map (<http://tangrams.github.io/cinnabar-style>) (view source on Github (<http://github.com/tangrams/cinnabar-style>))
- **More labels:** preview map (<http://tangrams.github.io/cinnabar-style-more-labels>) (view source on Github (<http://github.com/tangrams/cinnabar-style-more-labels>))
- **No labels:** preview map (<http://tangrams.github.io/cinnabar-style-no-labels>) (view source on Github (<http://github.com/tangrams/cinnabar-style-no-labels>))



- **Zinc**

- **Default:** preview map (<http://tangrams.github.io/zinc-style>) (view source on Github (<http://github.com/tangrams/zinc-style>))
- **More labels:** preview map (<http://tangrams.github.io/zinc-style-more-labels>) (view source on Github (<http://github.com/tangrams/zinc-style-more-labels>))

- **No labels: preview map** (<http://tangrams.github.io/zinc-style-no-labels>) (view source on **Github** (<http://github.com/tangrams/zinc-style-no-labels>))



Developer resources

So how do you actually use this stuff? Tangram styles are called “scenes” and are written in **YAML** (<https://en.wikipedia.org/wiki/YAML>), a human-friendly format that relies on indentation rather than delimiters (and is a superset of JSON). The scene file (e.g.: **zinc-style.yaml** (<https://github.com/tangrams/zinc-style/blob/gh-pages/zinc-style.yaml>)) requires a vector tile source, not the raster image tiles that you may be used to. To use Mapzen’s vector tile service you must first obtain a free developer API key and update your scene file with that key.

Sign up for a Mapzen Vector Tiles API key

Mapzen Vector Tiles are a free, shared tile service. As such, there are generous limitations on requests to prevent individual users from degrading the overall system performance.

1. Go to **<https://mapzen.com/developers>** (<https://mapzen.com/developers>).
2. Sign in with your GitHub account. If you have not done this before, you need to agree to the terms first.
3. Create a new key for Vector Tiles, and optionally, give it a project name so you can remember the purpose of the key.
4. Keep the web page open so you can copy the key into your source code later.

Building a Leaflet map using Tangram

Using Tangram and Mapzen’s vector tiles inside the popular **Leaflet** (<http://leafletjs.com>) mapping framework is easy. We’ll make it even easier soon to do this via a Leaflet **provider** (<https://github.com/leaflet-extras/leaflet-providers>), but in the meantime...

1. Update your copy of the scene file on **line 456** (<https://github.com/tangrams/zinc-style/blob/gh-pages/zinc-style.yaml#L456>) to reference the API key you created in Step 3 in the **Sign up** section above. url:

`https://tile.mapzen.com/mapzen/vector/v1/{layers}/{z}/{x}/{y}.{format}?api_key={api_key}`

2. Reference the **index-demo.html** (<https://github.com/tangrams/zinc-style/blob/gh-pages/index-demo.html>) file in any of the style repos for how to configure Leaflet with Tangram and the scene file (e.g.: **zinc-style.yaml** (<https://github.com/tangrams/zinc-style/blob/gh-pages/zinc-style.yaml>)).
3. Looking for a more sophisticated implementation that includes basic search? The main **index.html** ([index.html](#)) file has a more real world example.
4. Need help testing your map locally? See the README in each repo.
5. Wondering where to host your map? Make a public repo on Github (or fork ours!) and enjoy their **GitHub Pages** (<https://pages.github.com>) to github.io magic dance.

Tangram resources

1. **Procedural Cartography with Tangram** (<https://github.com/mapzen/presentations/tree/master/08-2015-JSGEO>) Patricio's presentation notes from **JS.Geo** (<http://www.jsgeo.com>) meetup last month.
2. **Walkthrough: Make a map with Tangram** (<https://mapzen.com/documentation/tangram/walkthrough/>) by Rhonda on Mapzen's documentation team.

You should be all set, happy mapping! But please let us know at **hello@mapzen.com** (<mailto:hello@mapzen.com>) or via Twitter **@mapzen** (<http://twitter.com/mapzen>) if you encounter any funk and we'll help you get up and running.

· 16 November 2015 ·



Nathaniel Vaughn Kelso

Map geek, cartographer, and data omnivore. Nathaniel leads the data team at Mapzen and vacations @nullisland.



Geraldine Sarmiento

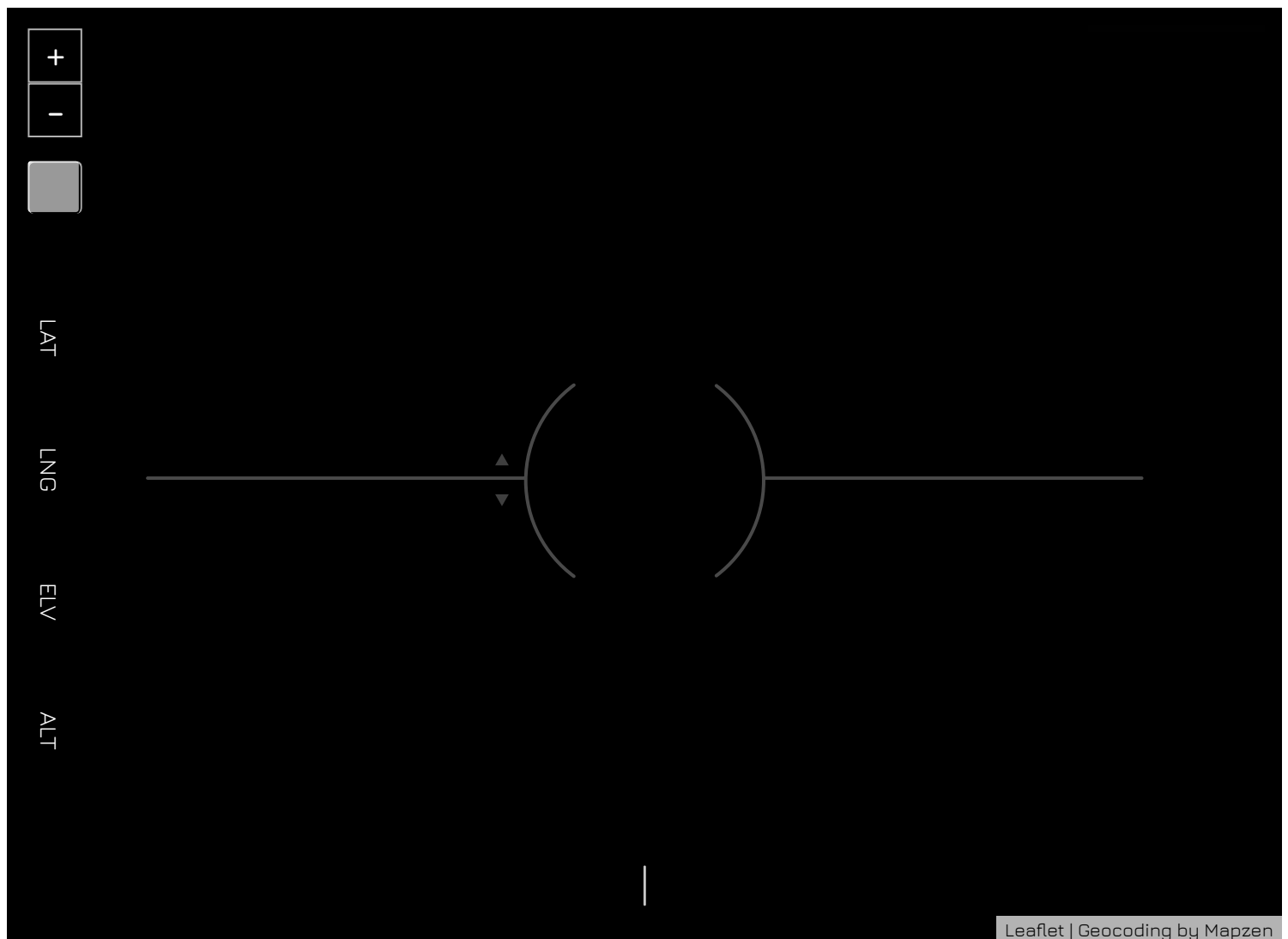
Geraldine is cartographer at Mapzen. She is addicted to shaders.

© 2017 Mapzen

Line of Sight

[tangram \(/tag/tangram\)](#) [demo \(/tag/demo\)](#)

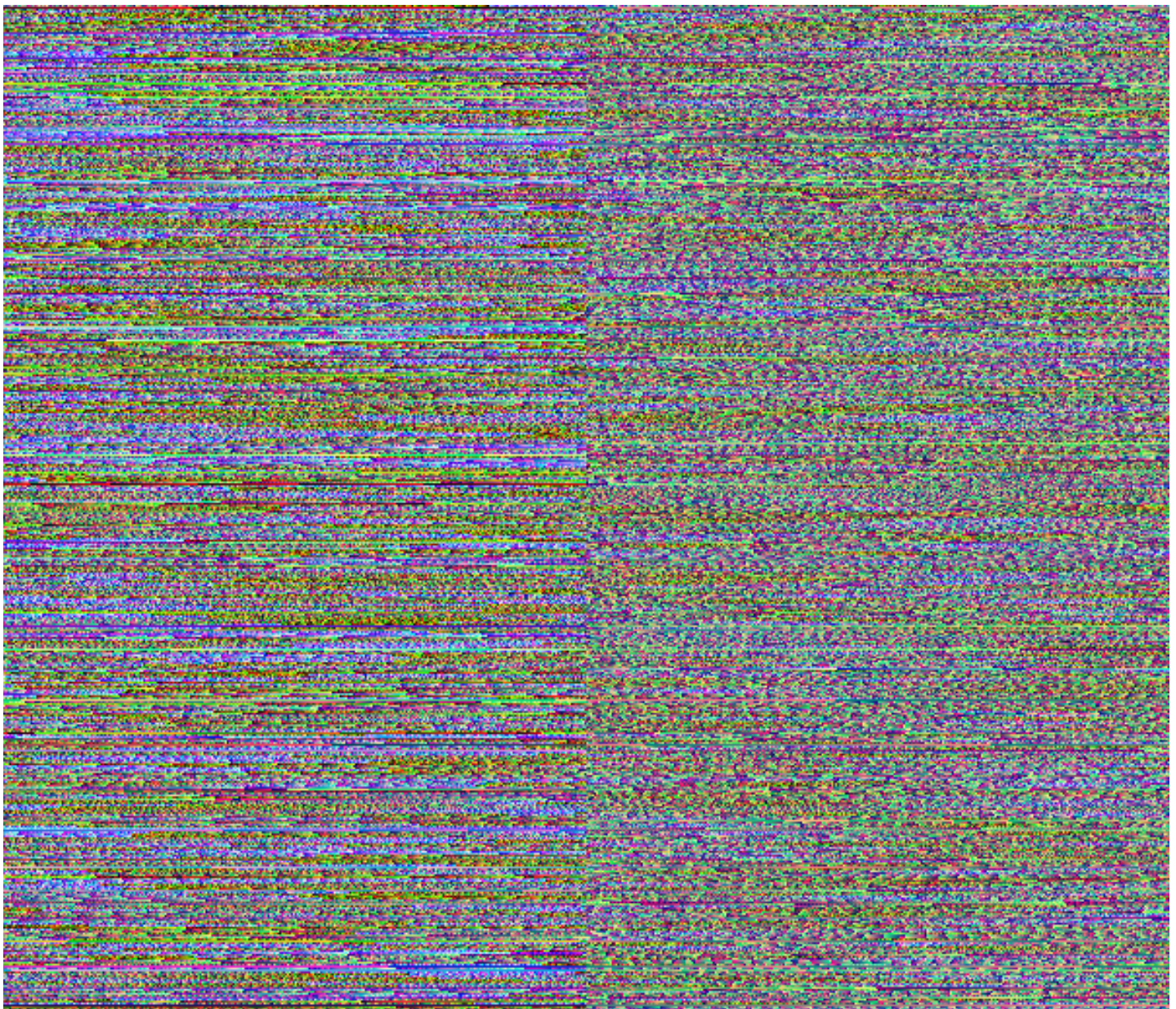
What satellites are above you right now? Patricio Gonzales Vivo wanted to find out, so he built **Line of Sight** (<https://patriciogonzalezvivo.github.io/LineOfSight>) using Tangram, a few Mapzen services, open orbital data and open source software libraries.



(This map is interactive! Open full screen ↗ (<http://patriciogonzalezvivo.github.io/LineOfSight>))

Patricio was looking into how use ham radio to communicate with satellites. The antennas have to be aimed, so he combined data from **SatNOGS** (<https://satnogs.org/>) (an open database of satellite transmitters) and **CelesTrak** (<http://www.celestrak.com/NORAD/elements/master.asp>) (a list of satellite codes), into GeoJSON using Python. He then used the **satellite.js library** (<https://github.com/shashwatak/satellite-js>) to pass satellite orbital data into Tangram.

Line of Sight uses **Mapzen Search** (<https://mapzen.com/projects/search>) and **Elevation** (<https://mapzen.com/documentation/elevation/elevation-service/>) to calculate the azimuth (direction) and elevation (angle) for a satellite over the next hour. Patricio decided to dynamically encode the locations of those satellites into image that can be quickly processed by Tangram. The longitude of 500 interesting, visible (and soon-to-be visible) satellites is encoded on the left side of the image, and their latitude is on the right.



Have a look at the code on **Github** (<https://github.com/patriciogonzalezvivo/LineOfSight>).

· 20 November 2015 ·

**Patricio Gonzalez Vivo**

Patricio is an artist and graphic engineer who speaks the language of light.

**John Oram**

Product marketing, burrito quality assurance, 4D map tiler. One day John will make as many maps as he writes about.

© 2017 Mapzen

Mapzen Acquires Mission Integers

FOR IMMEDIATE RELEASE

Mapzen announced today its acquisition of **Mission Integers** (<http://missionintegers.com/>), a San Francisco digital technology startup led by **Michal Migurski** (<https://twitter.com/michalmigurski>). The company provides artisanally crafted 64-bit integers as a service. Migurski will join Mapzen as part of the deal.

Mapzen CEO Randy Meech said: “Earlier this year we acquired **Brooklyn Integers** (<http://www.brooklynintegers.com/>), with **Aaron Cope** (<https://twitter.com/thisisaaronland>) joining Mapzen in June. **Who’s On First** (<https://whosonfirst.mapzen.com/>), our open gazetteer project, uses Brooklyn Integers to provide stable IDs for every location on earth. But because Brooklyn Integers mints only odd numbers, we soon realized an important part of the number space we’d overlooked. With this strategic acquisition Mapzen is doubling down on our artisanal integer strategy.”

“I am delighted to be an integral part of Mapzen,” commented Migurski. “They truly understand our overall mission and vision and will provide the resources to scale the service and realize its full potential to infinity, and beyond.” Mission Integers is currently said to be run on a Raspberry Pi in Migurski’s basement.

To put the deal in context, Etsy CTO **John Allspaw** (<https://twitter.com/allspaw>) commented from Brooklyn: “It’s not often that two systems of monotonously increasing sequences of numbers find each other in this crazy mixed-up world. Mission Integers and Brooklyn Integers truly complete each other.” **Nelson Minar** (<https://twitter.com/nelson>), San Francisco-based advisor to Mission Integers, showed an early grasp of the potential in a **blog post** (<https://nelsonlog.wordpress.com/2012/08/25/artisinal-integers-part-2/>) several years ago: “...should the demand for artisanal foundries explode and the number boffins need to start paying for their fixed gear bicycles and ironic bottled beers; simply start charging for those last few valuable foundry IDs. Scarcity is the business model’s best friend.”

Mapzen, a mapping company that uses only open data and software for all its products, is making an exception in the case of its combined integer service. Meech explained, “We believe that unique, auto-incrementing integers are the only part of the geo stack likely to retain any

proprietary value in the coming years. So we've decided to make an exception and keep both Brooklyn Integers and Mission Integers closed."

*Note: This post is (mostly) in jest. But **this part** (<http://mike.teczno.com/notes/1984-back-to-the-map.html>) is real.*

· 01 December 2015 ·



Randy Meech

Billerica Memorial High School graduate. BA, MTS. Mapzen CEO 2013-present.



Michal Migurski

Oakland-dwelling geodata cyclist.



Aaron Straup Cope

Aaron is happiest in the presence of olive oil.

© 2017 Mapzen

Targeted Editing – Where the streets have no names...

targeted-editing (/tag/targeted-editing) **osm** (/tag/osm)

Maps are current as of Oct 2016.

Should all streets have names? Actually, no...

Welcome to our **Targeted Editing** series – today you could search the Internet for a baby possum, or make **OpenStreetMap** (<http://www.openstreetmap.org/>) data awesome! (Both take about the same amount of time.)

Streets are one of the first things added in OpenStreetMap for any given area. There are always a number of segments that do not have names. In some cases, this is the reality – these may be little alleyways, paths, junctions, private roads, or even residential roads. Sometimes there are roads added that **do** have names, but the volunteer adding the data is digitizing the road from aerial imagery. More volunteers with **local knowledge** are needed to add attributes to these roads, like names, turn restrictions, speed limits, pavement type, and any number of road characteristics that are important to the local population.

Read on for some basic uses for road names, and see if you can use your **local knowlege** to add some where they are needed, while leaving the truly nameless roads alone!

How can street names be used?

1. **Help with search results.** If a street is made up of many segments and some of them have names, you can usually search and find the street. If all of the segments that make up the street have no name at all, you're out of luck!
2. **Help with turn-by-turn directions.** Routing is hard. There are a lot of features in the data that are necessary before you can generate great routes. There are also many features related to routing. One of them is spoken or listed turn-by-turn directions.

Which would you prefer?

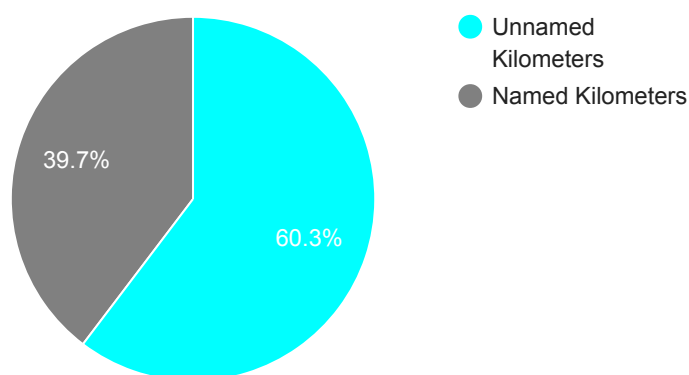
- a. "turn right in 500 feet at Elm Street"
- b. "turn right in 500 feet"
- c. "turn right in 500 feet at [pause of silence or blank space]"

3. **Help with map labeling.** The more you zoom in, the more labels you might see, but not if the features don't have names.

We are missing some street names here!

There are many different types of streets, but check out this pie chart for residential roads in Mexico City:

Mexico City, RESIDENTIAL Roads Without Names



We have some work to do!

How you can help improve streets without names:

Here is a map styled by **Peter Richardson** (<https://github.com/meetar>) & **Nathaniel V. Kelso** (<https://github.com/nvkelso>) showing many types of roads that don't have names. To add a name, hover over one of the bright blue highlighted roads to bring up an info bubble with links to editing tools that can be used to add the missing names!



Leaflet | Geocoding by Mapzen

(This map is interactive! Open full screen ↗ (<https://mapzen-data.github.io/targeted-editing/te-street-names/map/?roads>))

(If you find a road that appears named in our map, but is still highlighted, you may have found a duplicate feature. That's the topic of another post...)

Not familiar with Mexico City? Search or pan over to your home town to contribute your local knowledge to the map. You will see your changes right away in OpenStreetMap. You will also be able to see them in all versions of the **Mapzen Vector Tiles** (<https://mapzen.com/projects/vector-tiles>) within an hour, including the map right here on this page!

Need instructions on how to edit with iD? Here are some links to outstanding tutorials from LearnOSM, the OpenStreetMap wiki, and the United States Department of State's Humanitarian Information Unit:

- **LearnOSM** (<http://learnosm.org/en/>)
- **OSM Beginners' Guide** (http://wiki.openstreetmap.org/wiki/Beginners'_guide)

- **MapGive Learn to Map** (<http://mapgive.state.gov/learn-to-map/>)

Are you a mapping wiz and interested in a more advanced editor? Try out **JOSM** (<https://josm.openstreetmap.de>) with **excellent documentation** (<https://www.mapbox.com/blog/osm-mapping-guide/>) from Mapbox.

Thanks, and please check back soon for the next in our Targeted Editing series!

All the posts in the Targeted Editing series:

- **Airports** (<https://mapzen.com/blog/targeted-editing-airport-polygons>)
- **Banks** (<https://mapzen.com/blog/new-years-resolutions-money/>)
- **Building Names** (<https://mapzen.com/blog/targeted-editing-name-that-building>)
- **Campus Mapping** (<https://mapzen.com/blog/targeted-editing-campus-mapping/>)
- **Cycleways** (<https://mapzen.com/blog/targeted-editing-cycleways/>)
- **Fitness** (<https://mapzen.com/blog/new-years-resolutions-fitness>)
- **Groceries** (<https://mapzen.com/blog/new-years-resolutions-groceries>)
- **Hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>)
- **Schools** (<https://mapzen.com/blog/targeted-editing-school-polygons>)
- **Stadiums & Parking** (<https://mapzen.com/blog/targeted-editing-tailgate-mania>)
- **Street Names** (<https://mapzen.com/blog/targeted-editing-no-name-roads>)
- **Transit Colours** (<https://mapzen.com/blog/targeted-editing-transit-colours>)
- **Travel & Lodging** (<https://mapzen.com/blog/new-years-resolutions-travel>)

· 02 December 2015 ·



Indy Hurt

Indy was our resident data scientist lending her geographic expertise to all things "open".

An Open Letter to Mapzen's CEO

Hey Randy,

While I agree with the sentiment of **what you've written** (<https://mapzen.com/blog/an-open-letter-to-audi-ag-bmw-group-and-daimler-ag>) to the new owners of HERE, I can't help feeling there's a distinctly anti-OSM vibe from this piece. It seems like you're suggesting that HERE's data might be opened up to external contribution ("It would certainly reduce your ongoing data collection costs") as a competitor to OSM, which would only serve to fracture the open data community.

There's already several other projects in the global, open geographic data space; **Natural Earth** (<http://www.naturalearthdata.com/>), **openaddresses** (<http://openaddresses.io/>), etc... which receive attention from the global open data community, which arguably compete with OSM. Competition is, generally, good and healthy and increases diversity.

However, it seems to me that the point of the "open data revolution", and what we should be pushing for, is the opening up of a high quality, global geographic dataset and the growing irrelevance and devaluation of proprietary geographic data. What I think we need to be *very* careful of is what traditionally happens to the political Left; that despite lip service to "solidarity", they expend their effort fighting amongst themselves (as parodied by Monty Python's **Judean People's Front sketch** (https://www.youtube.com/watch?v=gb_qHP7VaZE)) and concentrating on the small differences between them rather than the large one separating them from the Right. One day, soon, when "open" has won, we can afford to split our communities and duplicate our efforts. Until then, all the time we're advocating a plurality of projects, we're helping proprietary data last a little bit longer.

Regardless of whether they actually *would* open up the data, and we've no reason to believe that they could or will, it sounded like you were advocating for the first true "competitor" to OSM. Which, at this stage, would only help Google / TomTom. You're right, it'll happen eventually, but I had hoped that Google, Apple, MQ, etc... would be using and contributing open data before then.

Cheers,

Matt

· 04 December 2015 ·



Matt Amos

OpenStreetMap developer-contributor and chilli enthusiast. Writes vector tile and other data-mastication tools at Mapzen.

© 2017 Mapzen

An "Open" Letter to Audi AG, BMW Group, and Daimler AG

osm (/tag/osm)

Congratulations on your **recent acquisition of HERE**

(<http://techcrunch.com/2015/12/04/nokia-closes-its-2-8b-sale-of-here-to-the-audi-bmw-and-daimler-car-consortium/>), formerly Nokia's mapping division. After the deal was announced last summer, our friend **Marc Prioleau** (<http://prioleuadv.com/archives/651>) wrote, "I hope they thrive. The world needs more options. Not fewer." Mapzen wholeheartedly agrees and wishes you the best.

Last June I **gave a talk at State of the Map** (<http://stateofthemap.us/osmba-the-history-and-future-of-companies-in-openstreetmap/>) at the UN about companies and open mapping. I pointed out the rapidly declining value of proprietary map data caused initially by Google Maps, but accelerated by the growing dominance of OpenStreetMap. Case in point: Nokia created HERE by the \$8.1 billion acquisition of NAVTEQ in 2007 and added to it with a number of smaller acquisitions. Eight years later, the value of the whole has decreased by over \$5 billion if you look at today's closing price of \$2.8 billion. This is a staggering loss of value. See **detailed earlier analysis** (http://www.gpsbusinessnews.com/Nokia-s-HERE-Value-is-Down-to-2-Billion_a5130.html) from GPS Business News.

Unfortunately, I believe that the value of the HERE asset will continue to drop. Multiple large Silicon Valley companies are investing heavily in this area, and a fast-growing group of startups allied around open mapping data will prove a formidable threat. You have acquired HERE as a consortium of three large auto companies: this may reduce risk and cost in the short term, but will it provide enough responsibility and innovation to ensure you can withstand these threats?

The heyday of map data as a defensible proprietary asset is rapidly coming to a close. One might compare the HERE dataset to the old Encyclopedia Britannica, with OpenStreetMap as its Wikipedia. If you proceed without strong action, this deal may wind up a similar footnote years from now.

Alternatively, you could do something bold: **open up the HERE dataset and let an open data community help** with its stewardship. The reason you would buy a mapping company like HERE is to secure long-term independence from Silicon Valley for your cars. You don't need to keep this data proprietary for that. In fact, it might be more advantageous for you to open it up as a way to compete with your rivals, who still operate as though this data has long-term proprietary value. It would certainly reduce your ongoing data collection costs. And you could still sell services on top of the data, as HERE currently does. And companies would likely join you to help with ongoing maintenance: Mapzen certainly would.

Just a suggestion of course, but I believe that you have an opportunity to use your recent acquisition to create something of permanent, durable value, and I encourage you to try it. This data will all be open one day anyway, and it would be great to work with you. Good luck!



image via **Nokia** (<http://company.nokia.com/sites/default/files/gallery/images/here-side-left-transparent.png>)

· 04 December 2015 ·

Randy Meech



Billerica Memorial High School graduate. BA, MTS. Mapzen CEO 2013-present.

© 2017 Mapzen

Targeted Editing – Does Your Hospital Have a Polygon?

targeted-editing (/tag/targeted-editing) **osm** (/tag/osm)

Maps are current as of Oct 2016.

For hospitals, polygons are twice as nice!

Welcome to our **Targeted Editing** series where today you can eat a peppermint, or make **OpenStreetMap** (<http://www.openstreetmap.org/>) data excellent! (Both take about the same amount of time.)

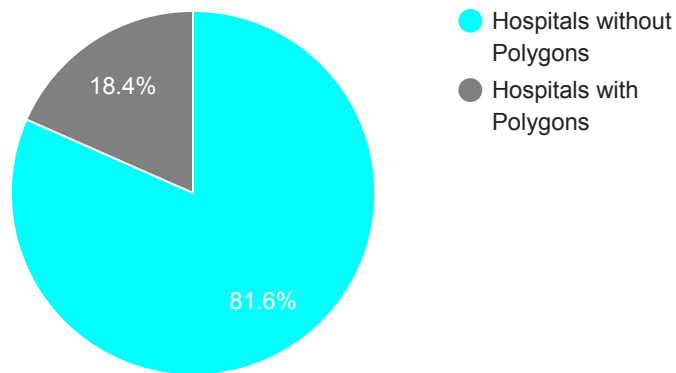
Read on for some basic uses for hospital polygons, and where you can check to see if you can use your **local knowlege** to add some polygons where they are needed.

How can hospital polygons be used?

1. **Help with map rendering.** No, this doesn't mean cook the fat out of it, this means help with how things are cartographically displayed on the map. A map label for a point is great to have, but a polygon for display is even better.
2. **Help with display order.** In many cases, the size of a hospital indicates significance. When hospitals have polygons, the size of the hospital can be used to determine how much you should zoom in to the map before it appears. Bigger hospitals can be shown sooner than smaller hospitals.
3. **Help with search rank.** When searching, a search engine can use a number of things to prioritize what features are returned. Some examples might be associated attributes like population, or even the physical size of a feature. Since points have no area, adding a polygon could be an excellent enhancement.

We are missing some hospital polygons here!

Check out this pie chart for hospitals in Los Angeles:

Los Angeles, Hospital Polygons

We have some work to do!

How you can help improve hospitals without polygons:

Here is a map styled by **Peter Richardson** (<https://github.com/meetar>) & **Nathaniel V. Kelso** (<https://github.com/nvkelso>) showing hospitals that don't have polygons. To add a polygon, hover over one of the bright blue highlighted hospitals to bring up an info bubble with links to editing tools that can be used to add the missing polygons.



Leaflet | Geocoding by Mapzen

(This map is interactive! Open full screen ↗ (<https://mapzen-data.github.io/targeted-editing/te-hospital-polygons/map/?hospitals>))

See the wiki pages for **amenity=hospital**

(<http://wiki.openstreetmap.org/wiki/Tag:amenity%3Dhospital>) for lots of details and a great example. A polygon tagged amenity=hospital should cover the grounds of the hospital, **not just the building footprint**.

(If you find a hospital that appears to have a polygon, but it's still highlighted, you may have found a duplicate or a building that was tagged amenity=hospital instead of the hospital grounds. Improving those is a topic for another post...)

Not familiar with Los Angeles? Search or pan over to your home town to contribute your local knowledge to the map. You will see your changes right away in OpenStreetMap. You will also be able to see them in all versions of the **Mapzen Vector Tiles** (<https://mapzen.com/projects/vector-tiles>) within an hour, including the map right here on this page!

Need instructions on how to edit with iD? Here are some links to outstanding tutorials from LearnOSM, the OpenStreetMap wiki, and the United States Department of State's Humanitarian Information Unit:

- **LearnOSM** (<http://learnosm.org/en/>)
- **OSM Beginners' Guide** (http://wiki.openstreetmap.org/wiki/Beginners'_guide)
- **MapGive Learn to Map** (<http://mapgive.state.gov/learn-to-map/>)

Are you a mapping wiz and interested in a more advanced editor? Try out **JOSM** (<https://josm.openstreetmap.de>) with **excellent documentation** (<https://www.mapbox.com/blog/osm-mapping-guide/>) from Mapbox.

Thanks, and please check back soon for the next post in our series!

All the posts in the Targeted Editing series:

- **Airports** (<https://mapzen.com/blog/targeted-editing-airport-polygons>)
- **Banks** (<https://mapzen.com/blog/new-years-resolutions-money/>)
- **Building Names** (<https://mapzen.com/blog/targeted-editing-name-that-building>)
- **Campus Mapping** (<https://mapzen.com/blog/targeted-editing-campus-mapping/>)
- **Cycleways** (<https://mapzen.com/blog/targeted-editing-cycleways/>)
- **Fitness** (<https://mapzen.com/blog/new-years-resolutions-fitness>)
- **Groceries** (<https://mapzen.com/blog/new-years-resolutions-groceries>)
- **Hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>)
- **Schools** (<https://mapzen.com/blog/targeted-editing-school-polygons>)
- **Stadiums & Parking** (<https://mapzen.com/blog/targeted-editing-tailgate-mania>)
- **Street Names** (<https://mapzen.com/blog/targeted-editing-no-name-roads>)
- **Transit Colours** (<https://mapzen.com/blog/targeted-editing-transit-colours>)
- **Travel & Lodging** (<https://mapzen.com/blog/new-years-resolutions-travel>)

· 07 December 2015 ·



Indy Hurt

Indy was our resident data scientist lending her geographic expertise to all things "open".

© 2017 Mapzen

Targeted Editing – Does Your School Have a Polygon?

targeted-editing (/tag/targeted-editing) **osm** (/tag/osm)

Maps are current as of Oct 2016.

For schools, polygons are twice as nice!

Welcome to our **Targeted Editing** series where today you can try to decide which fruit cake is edible, or make **OpenStreetMap** (<http://www.openstreetmap.org/>) data incredible! (Both take about the same amount of time.)

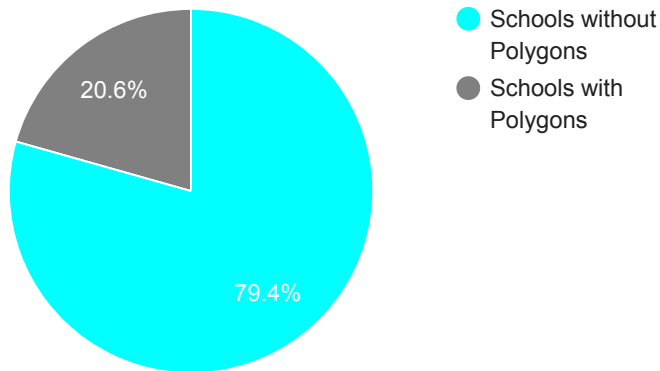
Read on for some basic uses for school polygons, and where you can check to see if you can use your **local knowlege** to add some polygons where they are needed.

How can school polygons be used?

1. **Help with map rendering.** Map rendering involves taking the raw data and turning it into a map. A map label is great to have, but a polygon is even better.
2. **Help with display order.** In many cases, the size of a school indicates significance. When schools have polygons, the size of the school can be used to determine how much you should zoom in to the map before it appears. Bigger schools can be shown sooner than smaller schools.
3. **Help with search rank.** When searching, a search engine can use a number of things to prioritize what features are returned. Some examples might be associated attributes like type, or even the physical size of a feature. Since points have no area, adding a polygon could be an excellent enhancement.

We are missing some school polygons here!

Check out this pie chart for schools in Atlanta:

Atlanta, School Polygons

We have some work to do!

How you can help improve schools without polygons:

Here is a map styled by **Peter Richardson** (<https://github.com/meetar>) & **Nathaniel V. Kelso** (<https://github.com/nvkelso>) showing schools that don't have polygons. To add a polygon for a school that is only represented by a point, hover over one of the bright blue highlighted school points to bring up an info bubble with links to editing tools that can be used to add the missing polygon.



Leaflet | Geocoding by Mapzen

(This map is interactive! Open full screen ↗ (<https://mapzen-data.github.io/targeted-editing/te-school-polygons/map/index.html?schools>))

See the wiki pages for **amenity=school**

(<https://wiki.openstreetmap.org/wiki/Tag:amenity%3Dschoo>) for lots of details and a great example. A polygon tagged `amenity=school` should cover the grounds of the school, **not just the building footprint**, while the school building itself should be tagged as `building=school` . This diagram from the OpenStreetMap Wiki gives a good overview of the tagging options.

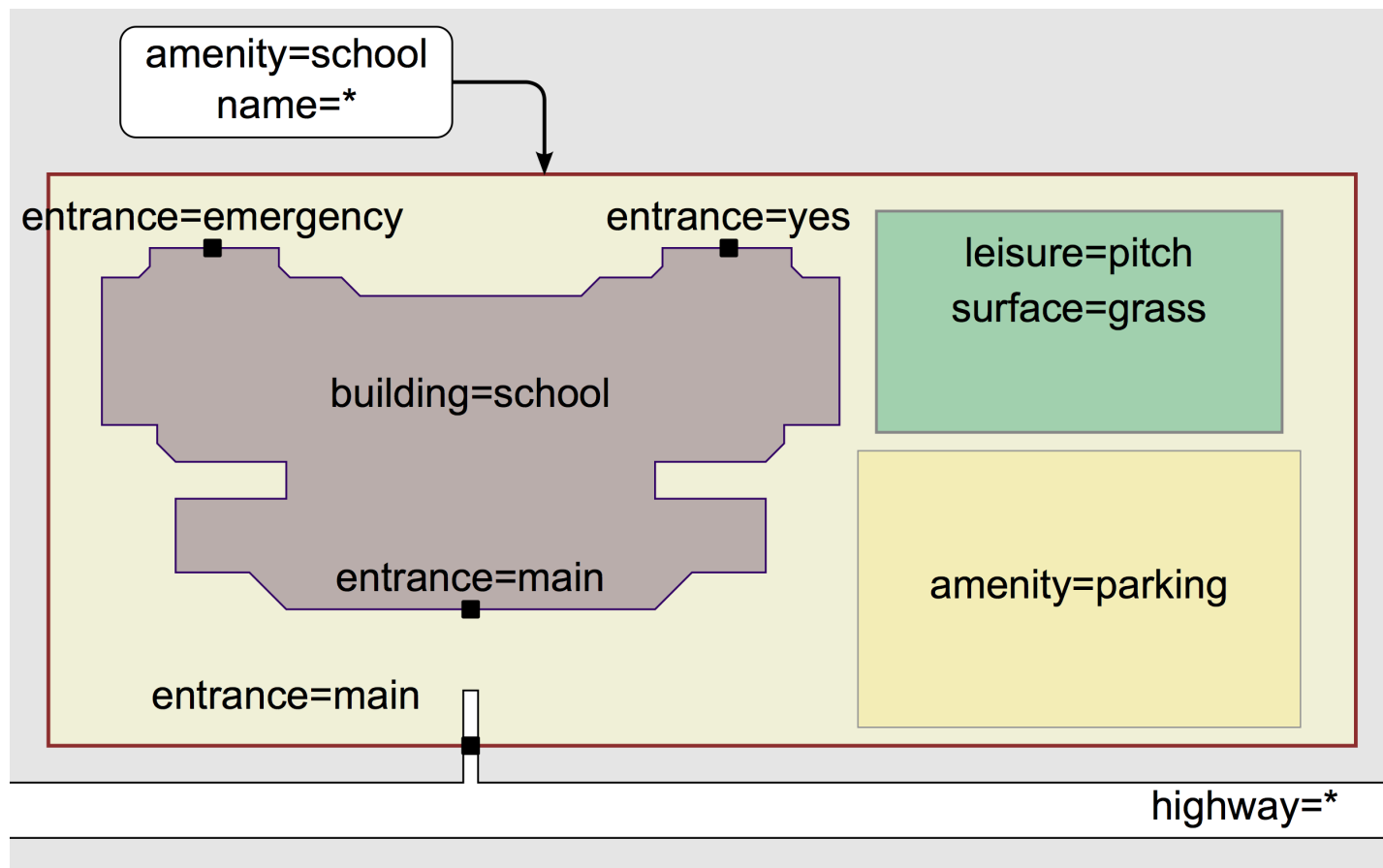


image via **OSM wiki** (<https://wiki.openstreetmap.org/wiki/Tag:amenity%3Dschoo>)

Including your sources with the **source** (<https://wiki.openstreetmap.org/wiki/Key:source>) key is incredibly helpful. This includes the imagery you use to trace features, and any authoritative websites as long as they allow their use for this purpose.

(If you find a school that appears to have a polygon, but it's still highlighted, you may have found a duplicate or a building that was tagged `amenity=school` instead of the school grounds. Improving those is a topic for another post...)

Not familiar with Atlanta? Search or pan over to your home town to contribute your local knowledge to the map. You will see your changes right away in OpenStreetMap. You will also be able to see them in all versions of the **Mapzen Vector Tiles** (<https://mapzen.com/projects/vector-tiles>) within an hour, including the map right here on this page!

Need instructions on how to edit with iD? Here are some links to outstanding tutorials from LearnOSM, the OpenStreetMap wiki, and the United States Department of State's Humanitarian Information Unit:

- **LearnOSM** (<http://learnosm.org/en/>)

- **OSM Beginners' Guide** (http://wiki.openstreetmap.org/wiki/Beginners'_guide)
- **MapGive Learn to Map** (<http://mapgive.state.gov/learn-to-map/>)

Are you a mapping wiz and interested in a more advanced editor? Try out **JOSM** (<https://josm.openstreetmap.de>) with **excellent documentation** (<https://www.mapbox.com/blog/osm-mapping-guide/>) from Mapbox.

Thanks, and please check back soon for the next in our series!

All the posts in the Targeted Editing series:

- **Airports** (<https://mapzen.com/blog/targeted-editing-airport-polygons>)
- **Banks** (<https://mapzen.com/blog/new-years-resolutions-money/>)
- **Building Names** (<https://mapzen.com/blog/targeted-editing-name-that-building>)
- **Campus Mapping** (<https://mapzen.com/blog/targeted-editing-campus-mapping/>)
- **Cycleways** (<https://mapzen.com/blog/targeted-editing-cycleways/>)
- **Fitness** (<https://mapzen.com/blog/new-years-resolutions-fitness>)
- **Groceries** (<https://mapzen.com/blog/new-years-resolutions-groceries>)
- **Hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>)
- **Schools** (<https://mapzen.com/blog/targeted-editing-school-polygons>)
- **Stadiums & Parking** (<https://mapzen.com/blog/targeted-editing-tailgate-mania>)
- **Street Names** (<https://mapzen.com/blog/targeted-editing-no-name-roads>)
- **Transit Colours** (<https://mapzen.com/blog/targeted-editing-transit-colours>)
- **Travel & Lodging** (<https://mapzen.com/blog/new-years-resolutions-travel>)

· 10 December 2015 ·



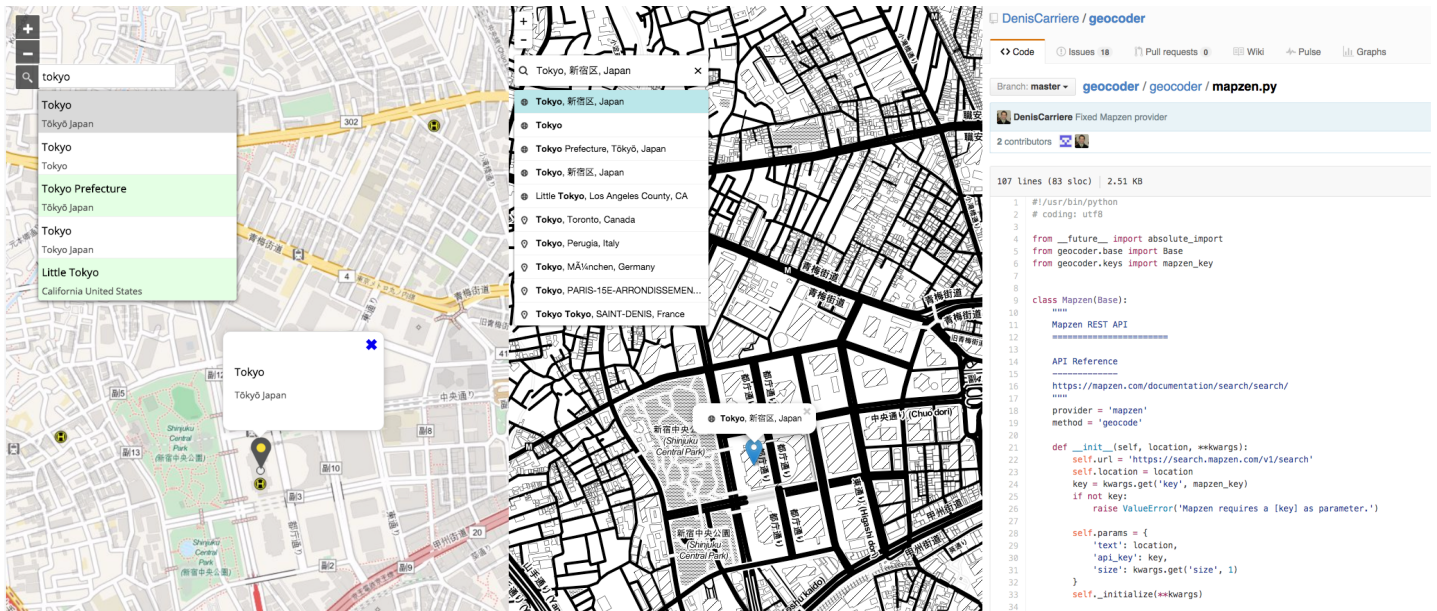
Indy Hurt

Indy was our resident data scientist lending her geographic expertise to all things "open".

Mapzen Search and Pelias Plugin Pulse

search (/tag/search)

We have four brief, but exciting pieces of news we wanted to share about our plugins and demos for Mapzen Search on the web.



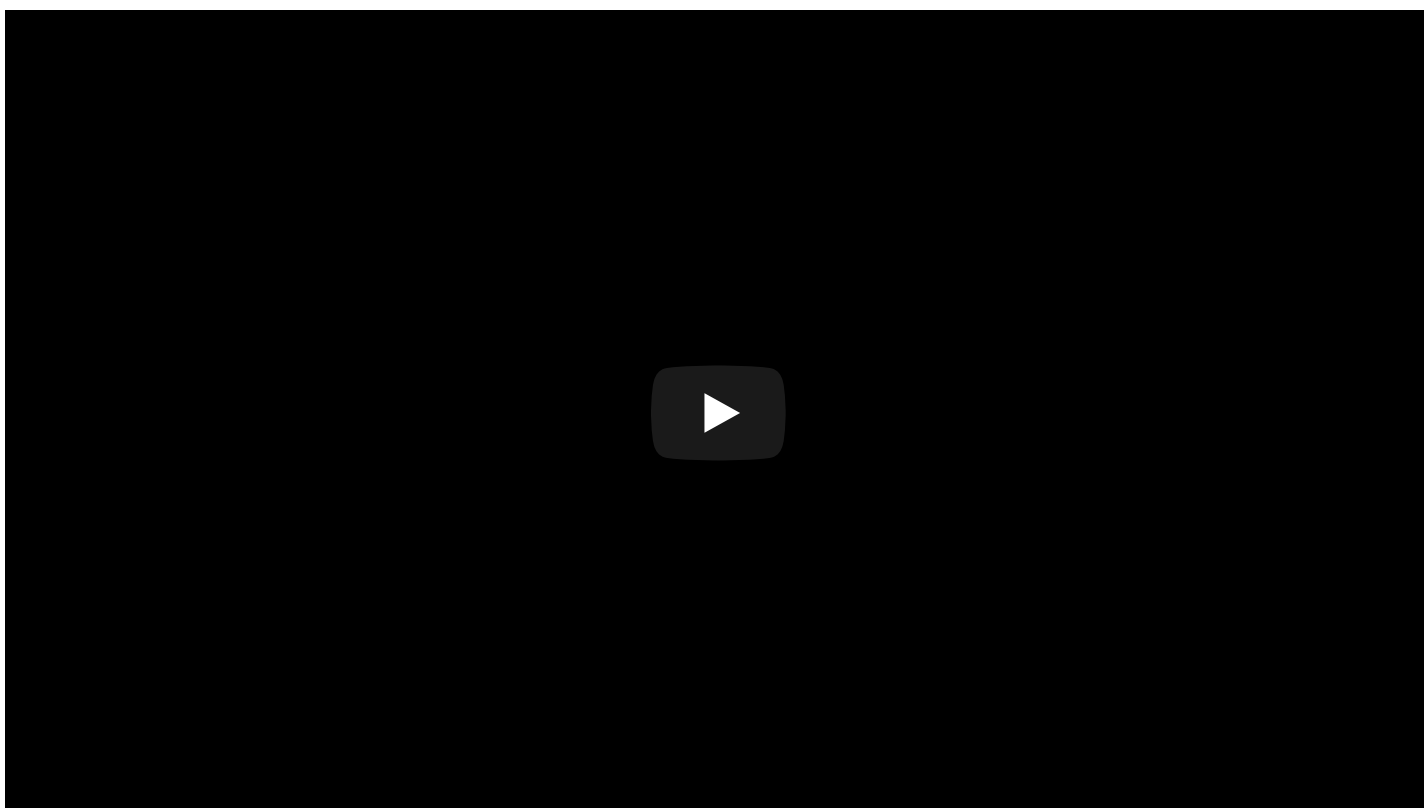
First, we're now supported by **Jonatas Walker's** (<https://github.com/jonataswalker>) geocoding plugin for OpenLayers, **ol3-geocoder** (<https://github.com/jonataswalker/ol3-geocoder>). If you're putting a map just about anywhere on the web, you should have a Mapzen Search as a drop-in geocoder, no matter your map display library. You can learn more about the **ol3-geocoder project** (<http://jonataswalker.github.io/ol3-geocoder/>) here, or take it for a spin with **this JSFiddle example** (<https://jsfiddle.net/9vjns06a/>) – be sure to add **your free Mapzen Search API Key** (<https://mapzen.com/developers>). Thanks **Jonatas** (<https://github.com/jonataswalker>)!

The second piece of news is for users of our existing **leaflet-geocoder** (<https://github.com/mapzen/leaflet-geocoder>) plugin for Mapzen Search and Pelias. We've moved this library into **Mapzen's Github** (<https://github.com/mapzen>) organization because our users were looking for it in relation to **Mapzen Search** (<https://mapzen.com/projects/search>). If you've installed the project via **npm** (<https://www.npmjs.com/package/leaflet-geocoder-mapzen>) or **Bower**

(<http://bower.io/search/?q=leaflet-geocoder-mapzen>), you'll need to update the package name in your dependency manifest to `leaflet-geocoder-mapzen`. Or, if you've installed it by downloading or hot-linking to the old location, you may want to update your link to the new location and filenames. Our package is also now hosted by **cdnjs** (<https://cdnjs.com/libraries/leaflet-geocoder-mapzen>)! You can learn more about the project name change in the **CHANGELOG** (<https://github.com/mapzen/leaflet-geocoder/blob/master/CHANGELOG.md#project-name-change>). Instructions on how to install it via cdnjs is now available on the **Mapzen Documentation site** (<https://mapzen.com/documentation/search/add-search-to-a-map/>). Thanks **Lou** (<https://mapzen.com/blog/find-your-community>)!

Also, **Denis Carriere** (<http://geocoder.readthedocs.org/>) added Mapzen Search to their Python-based **Geocoder** (<https://github.com/DenisCarriere/geocoder>). Instructions are available on their website and repo – Thanks **Denis (DenisCarriere)**!

Last but not least, **Ian Johnson**, aka **@enjalot** (<https://twitter.com/enjalot/status/675813341898510336>), made a **most excellent 20 minute tutorial** (<https://www.youtube.com/embed/XmwiZSopYQs>) on how to implement Mapzen Search!



Thank **Ian** (<https://twitter.com/enjalot>)!

If you're looking to search the world, **sign up for a developer key** (<https://mapzen.com/developers>) and take one of these for a spin!

· 14 December 2015 ·



David Riordan

Former product manager for Mapzen Search. Historical mapping, open tools, open access, and open data.

© 2017 Mapzen

Targeted Editing – Does Your Airport Have a Polygon?

targeted-editing (/tag/targeted-editing) **osm** (/tag/osm)

Maps are current as of Oct 2016.

For airports, polygons are twice as nice!

Welcome to our **Targeted Editing** series where today you can figure out how to cook with thyme, or make **OpenStreetMap** (<http://www.openstreetmap.org/>) data sublime! (Both take about the same amount of time.)

Read on for some basic uses for airport polygons, and where you can check to see if you can use your **local knowledge** to add some polygons where they are needed.

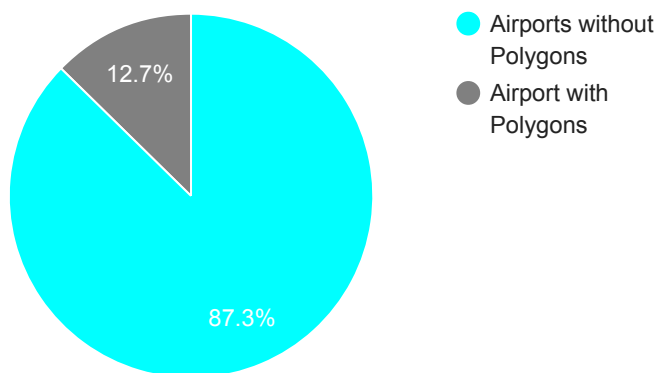
How can airport polygons be used?

1. **Help with map rendering.** No, this doesn't mean we need help adding a layer of sand and cement, this means help with how things are cartographically displayed on the map. A map label for a point is great to have, but a polygon is even better. It adds more detail and visual appeal.
2. **Help with display order.** In many cases, the size of an airport indicates significance. When airports have polygons, the size of the airport can be used to determine how much you should zoom in to the map before it appears. Bigger airports can be shown sooner than smaller airports.
3. **Help with search rank.** When searching, a search engine can use a number of things to prioritize what features are returned. Some examples might be associated attributes like type, or even the physical size of a feature. Since points have no area, adding a polygon could be an excellent enhancement.
4. **Help with routing.** An airport polygon helps identify the entire grounds of an airport; not just parking, terminal buildings, and runways. Airport polygons can help to indicate where driving might be restricted.

We are missing some airport polygons here!

Check out this pie chart for airports in the San Francisco Bay Area:

The Bay Area, Airports without Polygons



'Tis the season to travel, and regional airports need some love.

How you can help improve airports without polygons:

Here is a map styled by **Peter Richardson** (<https://github.com/meetar>) & **Nathaniel V. Kelso** (<https://github.com/nvkelso>) showing airports that don't have polygons. To add a polygon for an airport that is only represented by a point, click one of the bright blue highlighted airport points to bring up an info bubble with links to editing tools that can be used to add the missing polygons.



Leaflet | Geocoding by Mapzen

(This map is interactive! Open full screen ↗ (<https://mapzen-data.github.io/targeted-editing/te-airport-polygons/map/index.html?airports>))

See the wiki page for **aeroway=aerodrome**

(<http://wiki.openstreetmap.org/wiki/Tag:aeroway%3Daerodrome>) for lots of details.

Interested in adding even **more** airport features? Of course you are!! Visit the list of **aeroway**

(https://wiki.openstreetmap.org/wiki/Map_Features#Aeroway) features on the

OpenStreetMap wiki to help you learn how to digitize **runways**

(<https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Drunway>), **terminals**

(<https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Dterminal>), **gate numbers**

(<https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Dgate>), and many other useful details!

Including your sources with the **source** (<https://wiki.openstreetmap.org/wiki/Key:source>)

key is incredibly helpful. This includes the imagery you use to trace features, and any authoritative websites as long as they allow their use for this purpose.

If you find an airport that appears to have a polygon, but it's still highlighted, you may have found a duplicate, an airport with runways but not airport grounds polygon tagged aeroway=aerodrome, or some other interesting tagging variation. Improving those is a topic for another post...)

Not familiar with the San Francisco Bay Area? Search or pan over to your home town to contribute your local knowledge to the map. You will see your changes right away in OpenStreetMap. You will also be able to see them in all versions of the **Mapzen Vector Tiles** (<https://mapzen.com/projects/vector-tiles>) within an hour, including the map right here on this page!

Need instructions on how to edit with iD? Here are some links to outstanding tutorials from LearnOSM, the OpenStreetMap wiki, and the United States Department of State's Humanitarian Information Unit:

- **LearnOSM** (<http://learnosm.org/en/>)
- **OSM Beginners' Guide** (http://wiki.openstreetmap.org/wiki/Beginners'_guide)
- **MapGive Learn to Map** (<http://mapgive.state.gov/learn-to-map/>)

Are you a mapping wiz and interested in a more advanced editor? Try out **JOSM** (<https://josm.openstreetmap.de>) with **excellent documentation** (<https://www.mapbox.com/blog/osm-mapping-guide/>) from Mapbox.

Thanks, and please check back soon for the wrap-up of our series!

All the posts in the Targeted Editing series:

- **Airports** (<https://mapzen.com/blog/targeted-editing-airport-polygons>)
- **Banks** (<https://mapzen.com/blog/new-years-resolutions-money/>)
- **Building Names** (<https://mapzen.com/blog/targeted-editing-name-that-building>)
- **Campus Mapping** (<https://mapzen.com/blog/targeted-editing-campus-mapping/>)
- **Cycleways** (<https://mapzen.com/blog/targeted-editing-cycleways/>)
- **Fitness** (<https://mapzen.com/blog/new-years-resolutions-fitness>)
- **Groceries** (<https://mapzen.com/blog/new-years-resolutions-groceries>)
- **Hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>)
- **Schools** (<https://mapzen.com/blog/targeted-editing-school-polygons>)
- **Stadiums & Parking** (<https://mapzen.com/blog/targeted-editing-tailgate-mania>)
- **Street Names** (<https://mapzen.com/blog/targeted-editing-no-name-roads>)
- **Transit Colours** (<https://mapzen.com/blog/targeted-editing-transit-colours>)

- ***Travel & Lodging*** (<https://mapzen.com/blog/new-years-resolutions-travel>)

· 14 December 2015 ·



Indy Hurt

Indy was our resident data scientist lending her geographic expertise to all things "open".

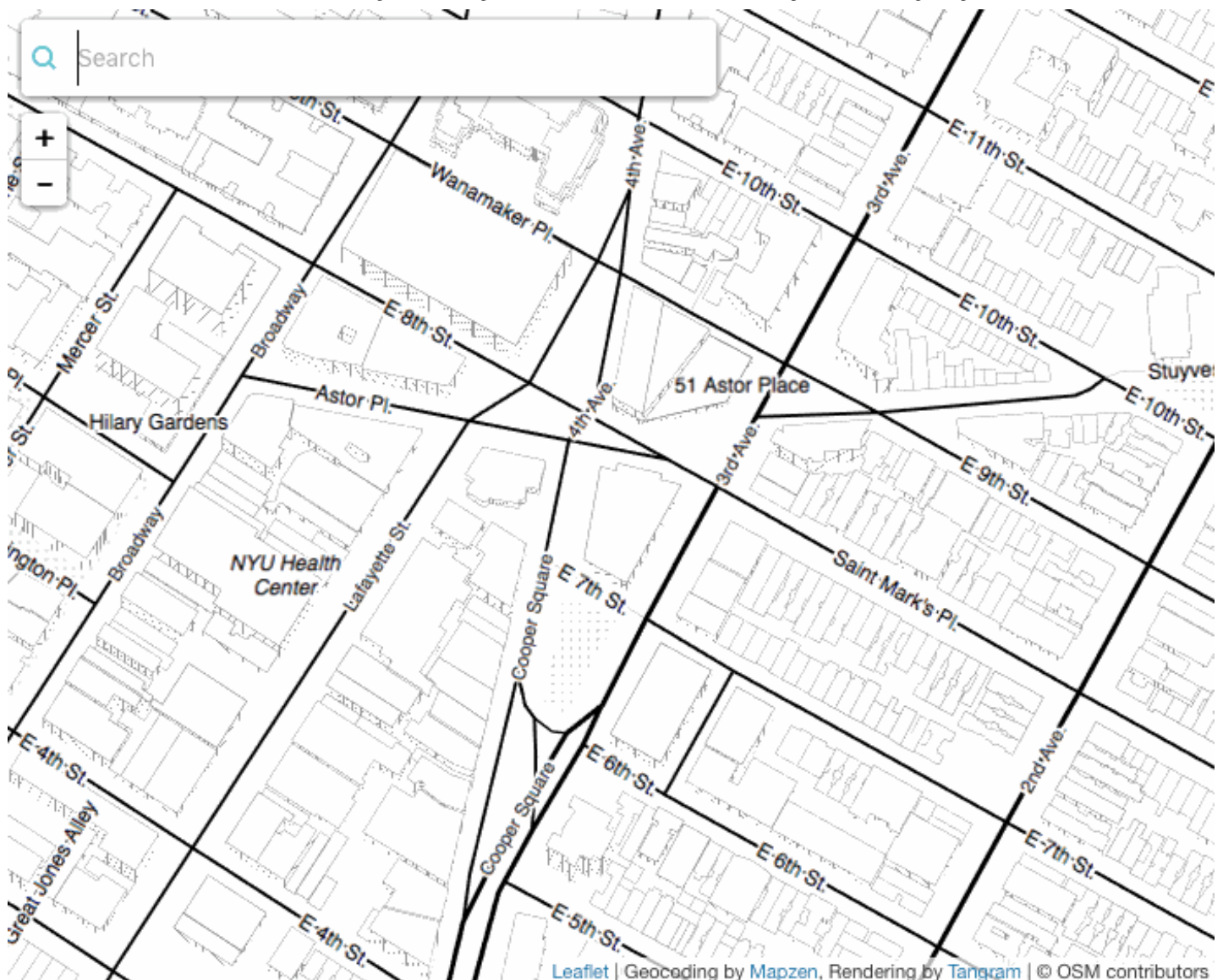
© 2017 Mapzen

Meaningful Geocoding – Address Search & The Two Core Principles of Geocoding

search (/tag/search) **geocoding** (/tag/geocoding)

There are many things involved in Searching for Places. It's not the easiest thing to do because, as it turns out, the world is a complex place.

*In our new series, **Meaningful Geocoding**, we're going to share the principles and practices of geocoding that we're incorporating as we build **Pelias** (<https://github.com/pelias/pelias>), our open source geocoder that powers our service **Mapzen Search** (<https://mapzen.com/projects/search>) entirely on open data.*



Searching for an address on 145th Ave. in Queens, NY **showing the streets 145th Ave, 145th Road, 145th Drive, right next to one another (<http://tangrams.github.io/cinnabar-style/#17/40.66296/-73.77541>)**.

There are a lot of ways that people search for places. Sometimes they search for places by name, like like a business or a landmark. But not every place has a name and names can be just about anything.

But nearly every place has an Address. They're long, they're complex, they're compound, and they can act as unique identifiers for a place. There are systems and patterns embedded in addresses providing incremental hinting about location, and we can use these patterns to return the best results, even if the addresses themselves **don't always make the most sense (<https://www.mjt.me.uk/posts/falsehoods-programmers-believe-about-addresses/>)**.¹

With that in mind, here are some principles that we're taking to heart as we build Mapzen Search and Pelias.

Heterogeneous and Unambiguous

When someone performs a search on a geocoder, the places that come back are ideally:

- Heterogeneous
- Unambiguous

Heterogeneous results match what the user entered for all matching places, big and small. That is, if the user only provides "New York" it's perfectly fine to return New York the city, New York the county, and New York the state.

Unambiguous results match what the user entered. In other words: don't return what the user didn't ask for. If the user entered "New York, NY", don't return "Village of New York Mills, Oneida County, NY" or "West New York, NJ".

Principles of Addressing

Users are, for the most part, rational individuals who provide input in a coherent fashion. Really! People's searches take the form of : **what** they're looking for, **where**.

The *where* is almost always organized hierarchically. That is, users enter

30 W 26th St, New York, NY 10010 ,

not

St NY W 30 10010 York New 26th

or

30 W 26th St, New York, 10010 NY .

Format details vary by country, but after centuries of addressing correspondence, societies worldwide have done a pretty good job of establishing standards so that addresses are interpreted one way. ²

On occasion, users (especially on mobile devices) leave out certain bits of information for the sake of efficiency but, even then, they will not rearrange the order. For example, I might enter an address as 30 W 26th St 10010 , leaving out the city and state.

Precision Geocoding

Address points are like connective tissue for geodata. Businesses are at addresses. People's homes are at addresses. They don't live in latitudes and longitudes, but that's where all our geo tools live. Because of this, precise full address geocoding is the bread and butter of commercial place search, and not to mention necessary to support end-user navigation. If an address isn't searchable in the system, someone can't drive there. Let's do our damndest to get them there.

Every geocoder can only be as good as the data behind it; it needs a model of the world to tell you where you're looking for. An address geocoder requires a lot of data that someone went out and collected.

There are two ways of precision geocoding street addresses and they're defined by the data they use: *point geocoding* and *interpolated geocoding*. They both require a lot of data, but point geocoding only works when the source data already contains an exact point matching the address, which is a monumental effort to collect. ³

Conversely, interpolated geocoding is used when the source data contains just the line segments of the road network and corresponding address number (block) ranges. It's usually a lot easier to use an existing map of the road network containing the start and stop segments for addresses than it is to go out and collect each and every address point. In this case, an interpolation geocoder will find the appropriate street and make a very educated guess at where along the line the house number resides.

When data is available, most geocoders implement both approaches, using point data when possible and falling back to interpolation when needed.

Known Address Data Points

Users often enter a full street+city+state with a known point address number that must be resolved as unambiguously as possible.

Example: 30 W 26th St, New York, NY

In this case, the address number is a known point (with a latitude/longitude) in the data.

A geocoder should return:

- 30 W 26th St, New York, NY

Because we can be certain this is an exact match, no other results should be returned, including:

- no other address numbers on W 26th St in New York
- no '30 W 26th St' in another city
- no '30 E 26th St, New York, NY'
- no businesses on W 26th St, New York, NY

Interpolated Addresses

Users can enter an address number which is unknown to the system but still falls within the known block ranges of the source data. The input is most likely valid, but our address point data doesn't have the exact house number.

Example: 87 W 26th St, New York, NY

In this case the address number is not a known address number in the point data, so the lat/lon returned is interpolated along the known line segments. The same rules apply as before: don't return anything else but the unambiguous address.

A geocoder should return:

- 87 W 26th St, New York, NY

Address Beyond Known Range

Users can enter an unknown house number that's outside all known block ranges in the source data.

Example: 99999 W 26th St, New York, NY

In this case there is no address number 99999 and the block ranges don't go that high. There are two ways to deal with this scenario:

- return the last block range for W 26th St, New York, NY

or:

- return the middle of the line segments for W 26th St, New York, NY representing the street in general, letting the user know that you're only returning the street because the house number wasn't found.

Geocoding Streets

Users sometimes don't know a house number so they just enter the street.

Example: W 26th St, New York, NY

There are several ways to deal with this. One approach is to return all the block ranges available in data:

1. [1-99] W 26th St, New York, NY
2. [100-199] W 26th St, New York, NY
3. etc.

Alternatively, the result returned can be just the lat/lng of the midpoint of the line representing the street.

Partial Street Geocoding

While rare (at least in the US/CA), it's possible that a street input can be ambiguous within a city. This normally occurs when there are multiple streets within the same city with different street types.

New York is notorious for partial street geocodes ⁴.

Example: 30 26th St, New York, NY

In this example, the user didn't supply a directional, like East 26th Street or West 26th Street . When directionals are present in the data but not supplied by the user, a geocoder should return results on all matching streets:

- 30 W 26th St, New York, NY
- 30 E 26th St, New York, NY

If, however, the supplied house number is only applicable to one or the other, a geocoder should only return the address that's physically possible. For example, 601 is within the known block ranges of W 26th St but not E 26th St (geographically this address would be in the East River).

Example: 23 10th St, NYC

In this example, the user didn't supply a directional or a borough, but implies that the address is within New York City (New York City is comprised of five smaller "boroughs", which addresses fall within). In this case there are multiple 10th Streets in Manhattan (East and West), as well as within the other boroughs. When there are multiple matching areas contained within the search and valid address ranges for all of them, a geocoder should return:

- 23 E 10th St, New York, NY
- 23 W 10th St, New York, NY
- 23 10th St, Brooklyn, NY
- 23 10th St, Staten Island, NY
- 38-23 10th St, Queens, NY ⁵

Example: 23 10th, New York, NY

In this example, the user supplied neither the directional nor the street type. A geocoder should return:

- 30 10th Ave, New York, NY
- 30 E 10th St, New York, NY
- 30 W 10th St, New York, NY

There is room for flexibility in this example for business rules to help make the decision. For example, if there is point data for some of the addresses but not others, it's perfectly acceptable to only return results for which there is point data.

What's Next

Addresses aren't the only places people search for. In the next installment we'll examine some of the rules for when people search for towns, states, or countries, and how to give unambiguous results, even when the search could mean many things.

If cracking the code of how we organize the world sounds interesting to you, we'd love to work with you. Our project is **100% open** (<https://github.com/pelias>), so it would be great to have you as an open source contributors, and **we're hiring** (<https://mapzen.com/jobs/search-engineer-node/>) for another person to join our Search team to work on geocoding full time.

1. Of course, searching for named places can take advantage of geocoding, and geocoding can use search technologies. This is what we we're working on to craft a scalable approach that helps solve both of these problems. ↩
2. For instance, Germany's addresses follow the pattern of "*StreetName HouseNumber*". Even still, that structure can be parsed out and helps localize results further. A thorough analysis of log files for any geocoder would find inputs that don't match commonly-accepted patterns this but such instances are extremely rare. ↩
3. In fact, collecting point address data is part of the genesis of **Google's Street View** (http://readwrite.com/2009/10/12/google_maps_ditches_teleatlas_in_favor_of_street_view_cars_crowdsourcing). But you don't have to drive every street to get good address point data. While it takes a lot of effort to create and collect point address data, there's a remarkable open data project **OpenAddresses** (<https://openaddresses.io>) which is building a massive directory of openly-available and -licensed address points dataset worldwide. If you like to hack on open data, helping contribute to OpenAddresses by hunting down and cleaning up open address data makes for a lot of fun and a big difference. ↩
4. Addresses in NYC can be ludicrous for a whole host of reasons. There are directional prefixes (north, south, east, and west). There are streets with the same name and valid addresses in different boroughs. We'll be looking into how to deal with the weird addresses in one of the upcoming posts of this series. ↩
5. The borough of Queens, NY has a whole host of **address anomalies** (<http://queens.brownstoner.com/2012/07/how-to-make-sense-of-the-street-addresses-and-grid-in-queens/>), starting with hyphenated addresses. The unique housenumber in this example is 23, but because the cross-street is 38th ave, the full address is 38-23 10th St, Queens, NY. ↩

· 16 December 2015 ·

**Stephen Hess**

Stephen works on geocoding exclusively as a means to fund his passion for designing and building wooden frames for old maps.

**David Riordan**

Former product manager for Mapzen Search. Historical mapping, open tools, open access, and open data.

© 2017 Mapzen

The Matrix – Mapzen's Time-Distance Service is Here!

Mapzen is pleased to announce the new Time-Distance Matrix service! Rather than issue a large number of individual routing requests, times and distances can now be efficiently computed to, from, and amongst points, for any costing method or travel mode available in Mapzen Turn-by-Turn (auto, bicycle, or pedestrian).

While a Time-Distance Matrix is usually thought of in terms of delivery and logistics, you may not realize just how useful it can be:

- Hungry for lunch? The Matrix can help you decide which pizza place is closest.
- Going house hunting? The Matrix can compute commute distances and times from potential houses to your office.
- Getting married or having a big party? The Matrix can provide times and distances to guests coming in from different airports.
- Running a bunch of errands around town? The Matrix can give you an idea of time between all your stops.

Wouldn't it be nice to quickly and easily calculate time and distance, all in one API call? Mapzen's Time-Distance Matrix service is here and will make your planning needs easier.

How does it work?

We provide three different Time-Distance Matrix request options. `one_to_many` is the first, and likely to be the most commonly used. `one_to_many` will return a row vector of the computed time and distance from the first (origin) location to each additional location provided. This could be useful for finding a place to eat lunch with your co-workers, knowing that you have to be back for that 1 o'clock meeting.

The second matrix type that we provide is the `many_to_one`. This request will return a column vector of the computed time and distance from each location to the last (destination) location provided. Someone planning their wedding or party can provide their guests the time and distance from nearby airports they could fly into and hotels where they could stay. Your wedding guests will definitely appreciate `many_to_one` !

Finally, we provide a matrix type of `many_to_many`. The most computationally intensive, `many_to_many` will return a square matrix of computed time and distance from each location to every other location. This could be useful for a delivery worker or traveling salesman planning their route.

The service will return your response for distance in either kilometers or miles, whichever units you request.

You can read the **Time-Distance Matrix documentation** (<https://mapzen.com/documentation/matrix>) for more information about the options and **sign up for an API key** (<https://mapzen.com/developers/>).

A small note on performance and limitations


Performance is fairly similar for `one_to_many` and `many_to_one` matrix requests. Technically, `many_to_one` expands from the destination along reverse paths until it reaches each origin. Therefore, it is slightly more expensive to do reverse paths than forward paths, but not noticeable. `many_to_many` does slow down performance, as it needs to compute every variation of time and distance from each location to every other location. We currently have a few location and distance limits in place for this purpose and you can read more about them in the **documentation** (<https://mapzen.com/documentation/matrix>). To keep the service available to everyone, API requests are rate limited to 2 queries per second and 5000 queries per day.

Examples

The following cases illustrate a few use cases of the Time-Distance Matrix service.

One_to_Many

Maybe you want to check out a local pizza joint that is near your office for lunch. Do I have time to go from my office (location 0) and Rocco's Pizza (location 5) that everyone's talking about?

 **MAPZEN Time-Distance Matrix**

Select an environment

production ▼

1. Select a matrix type

One-to-Many

Many-to-One

Many-to-Many

2. Add start point(s)

0

40.744290 , -73.990286

Click on the map to add a point

3. Add end point(s)

1

40.744355 , -73.989079

2

40.740510 , -73.987362

3

40.744323 , -73.992271

4

40.745144 , -73.994755

5


40.742453 , -73.997523


6


40.744339 , -73.984551

Click on the map to add points


Clear All








VIEW MATRIX





 **MAPZEN Time-Distance Matrix**

< Back

| From | To | Time (secs) | Distance (mi) |
|------|----|-------------|---------------|
| 0 | 1 | 127 | 0.111 |
| 0 | 2 | 418 | 0.359 |
| 0 | 3 | 159 | 0.14 |
| 0 | 4 | 342 | 0.301 |
| 0 | 5 | 647 | 0.567 |
| 0 | 6 | 469 | 0.408 |


Show rows: 10 ▼

Results: 1 – 6 of 6



Many_to_One

You're getting your wedding invitations ready to go out to your guests and you want to give them an idea of the times and distances that they will have to travel to the event (location 4) from their various city locations.

 **MAPZEN Time-Distance Matrix**

Select an environment
production ▼

1. Select a matrix type

One-to-Many

Many-to-One

Many-to-Many

2. Add start point(s)

0 39.973437 , -76.723366

1 40.280573 , -76.892624

2 40.337124 , -75.915527

3 38.903858 , -77.034760


Click on the map to add points


3. Add end point(s)


4 39.873912 , -76.983948

Click on the map to add a point

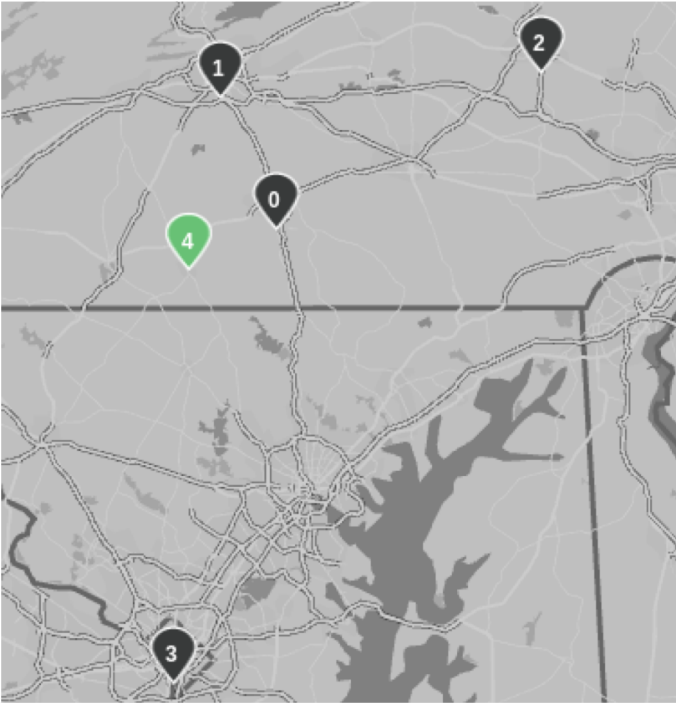
Clear All








VIEW MATRIX



 **MAPZEN Time-Distance Matrix**

< Back

| From | To | Time (secs) | Distance (mi) |
|------|----|-------------|---------------|
| 0 | 4 | 1340 | 18.206 |
| 1 | 4 | 2815 | 37.789 |
| 2 | 4 | 4528 | 73.309 |
| 3 | 4 | 6328 | 94.169 |

Show rows: 10 ▼ Results: 1 – 4 of 4

Many_to_Many

A driver in San Francisco has a list of deliveries to complete Monday morning. The warehouse is at location 0 and the driver wants to get list of times and distances from each location to every other location. This information can help the driver decide on the most optimized path as well as estimate overall travel time.

MAPZEN Time-Distance Matrix

Select an environment
production

1. Select a matrix type

One-to-Many Many-to-One **Many-to-Many**

2. Add start point(s)

0 37.747168 , -122.395935

1 37.745336 , -122.421083

2 37.738820 , -122.450609

3 37.751376 , -122.433100

4 37.752937 , -122.420826

5 37.758841 , -122.420912

6 37.760062 , -122.418766

7 37.791541 , -122.392759

8 37.778585 , -122.410269

9 37.790048 , -122.435846

10 37.798187 , -122.455416

11 37.802935 , -122.430010

12 37.800629 , -122.410784

13 37.798662 , -122.397823

14 37.758909 , -122.388811




15 37.730675 , -122.384262

16 37.725719 , -122.421856

17 37.757144 , -122.490692

18 37.739906 , -122.499275

Click on the map to add points

Clear All    **VIEW MATRIX**

Leaflet | Tangram | © OSM contributors | Mapzen, Geocoding by Mapzen

The sample images here were created using our **Test Utility** (<http://valhalla.github.io/demos/matrix/index.html>), which uses the Time-Distance Matrix service. Try it with your own examples!

Check it out!

Reach out (<http://twitter.com/ValhallaRouting>) if you have questions or suggestions! Also, look for optimized routing (including the classic “traveling salesman” problem) coming soon. Sign up for an API key on the **Mapzen Developers page (<https://mapzen.com/developers/>)** and **take a look at the documentation (<https://mapzen.com/documentation/matrix>)**. You can also try out our open-source routing with **Mapzen Turn-by-Turn (<https://mapzen.com/projects/valhalla>)**.

· 18 December 2015 ·



Kristen DiLuca

Kristen is a software engineer specializing in our routing API services. She also enjoys dabbling in javascript for our open source routing test tools and always welcomes new challenges.

© 2017 Mapzen

Targeted Editing Holiday Hiatus

targeted-editing (</tag/targeted-editing>) **osm** (</tag/osm>)

Maps are current as of Oct 2016.

Pull up a fireplace (<http://wiki.openstreetmap.org/wiki/Key:fireplace>) and a nice mug of eggnog to toast our Targeted Editing series this holiday season.



fire courtesy of **Virtual Fireplace** (<http://www.virtual-fireplace.net>)

Traveling by plane, train or automobile? Revisit our **airport polygons** (<https://mapzen.com/blog/targeted-editing-airport-polygons>) and **unnamed roads** (<https://mapzen.com/blog/targeted-editing-no-name-roads>) posts. Visiting home for the holidays? Use your trip down memory lane to enhance OpenStreetMap's representation of your old elementary **school** (<https://mapzen.com/blog/targeted-editing-school-polygons>). Holiday decorating accidents rise sharply at this time of year, but we are wishing you a safe season with no visits to **hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>), unless you are looking for things to map, of course.

While we have done our best to highlight some features that would greatly benefit from a quick edit, you may have found lots of additional things that need an enhancement or two. Revisit your favorite theme in the series to see where you can lend additional local knowledge:

- **Streets**

- Yes, names are important, but so are **turn restrictions** (<https://wiki.openstreetmap.org/wiki/Relation:restriction>), **speed limits** (<https://wiki.openstreetmap.org/wiki/Key:maxspeed>), the presence of **sidewalks** (<https://wiki.openstreetmap.org/wiki/Key:sidewalk>), **bike lanes** (<https://wiki.openstreetmap.org/wiki/Tag:highway%3Dcycleway>), and so many interesting details that you might know about the roads you travel regularly.

- **Hospitals**

- If you thought hospital polygons were useful, imagine all of the other hospital related features you could add. Some of our favorites include **buildings** (<https://wiki.openstreetmap.org/wiki/Tag:building%3Dhospital>) and their **entrances** (<https://wiki.openstreetmap.org/wiki/Key:entrance>), **emergency** (<https://wiki.openstreetmap.org/wiki/Key:emergency>) services, **parking areas** (<https://wiki.openstreetmap.org/wiki/Tag:amenity%3Dparking>) and **parking aisles** (https://wiki.openstreetmap.org/wiki/Tag:service%3Dparking_aisle).

- **Schools**

- Schools seem to corner the market on fun things to map. Add **playing fields** (<https://wiki.openstreetmap.org/wiki/Tag:leisure%3Dpitch>) and the types of **sports** (<https://wiki.openstreetmap.org/wiki/Key:sport>) played on them. Add **playgrounds** (<https://wiki.openstreetmap.org/wiki/Tag:leisure%3Dplayground>) and their **slides, swings, and jungle gyms** (<https://wiki.openstreetmap.org/wiki/Key:playground>)!

- **Airports**

- Airports are also comprised of loads of features. Why do you get to the airport two + hours before your flight? Because you want to map the **terminals** (<https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Dterminal>), **gates** (<https://wiki.openstreetmap.org/wiki/Tag:aeroway%3Dgate>), **shops** (<https://wiki.openstreetmap.org/wiki/Key:shop>) and **restaurants** (<https://wiki.openstreetmap.org/wiki/Tag:amenity%3Drestaurant>), of course.

Until next time, have a happy holiday season, and we hope you are looking forward to the continuation of our series in the new year!

All the posts in the Targeted Editing series:

- **Airports** (<https://mapzen.com/blog/targeted-editing-airport-polygons>)
- **Fitness** (<https://mapzen.com/blog/new-years-resolutions-fitness>)
- **Groceries** (<https://mapzen.com/blog/new-years-resolutions-groceries>)
- **Hospitals** (<https://mapzen.com/blog/targeted-editing-hospital-polygons>)
- **Schools** (<https://mapzen.com/blog/targeted-editing-school-polygons>)
- **Stadiums & Parking** (<https://mapzen.com/blog/targeted-editing-tailgate-mania>)
- **Streets** (<https://mapzen.com/blog/targeted-editing-no-name-roads>)

· 21 December 2015 ·



Indy Hurt

Indy was our resident data scientist lending her geographic expertise to all things "open".

© 2017 Mapzen